ファミコン大作戦

地球防衛軍がきみにおくる

# ファミリーベーシック™ ホップ

ファミコンでベーシックを
おぼえちゃえ
きみなら できる 挑戦だ！




地球防衛軍 編著
司令　谷岡康則

誠文堂新光社

ファミコン大作戦 ファミリーベーシック™ ホップ
誠文堂新光社

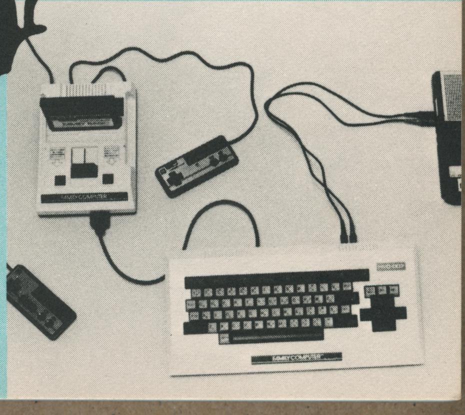
# ファミコン大作戦

# ファミリーベーシック

## ホップ





# HOP STEP

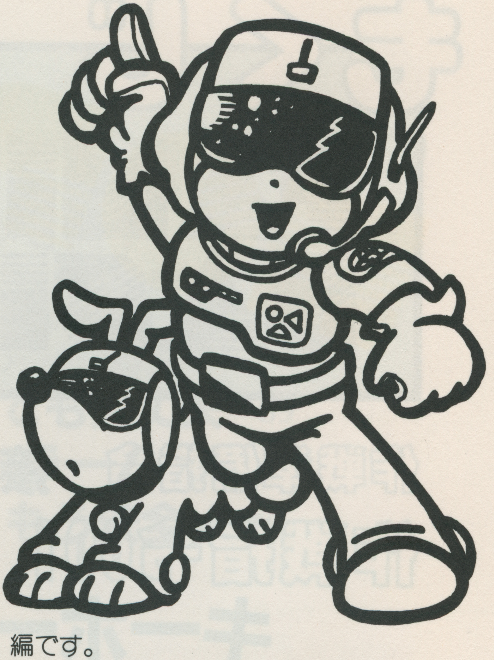
私たちのまわりには、「マイコン内蔵」や「コンピューター制御」と書かれた車や冷蔵庫、クーラー、果ては炊飯器などの商品があふれています。でも、それらの商品の中でコンピューターがどんな働きをしているのか、マイコンっていったい何なのか、わかるはずもありません。

ファミコンも、発売当初はそうでした。カセットを入れて、電源を入れる。コンピューターのことなんかわからなくても、おもしろいゲームが楽しめる、ゲームマシンでしかありませんでした。ところが、GAME BASIC カセットとキーボードが発売されたことで、ファミコンは単なるゲームマシンから、本当の意味でのファミリー・コンピューターへと大変身したのです。

すべての仕事をコンピューターに任せてしまって、人間にはコンピューターが何をしているかわからない社会にしてはいけません。人間が何かを考え、作り出す時、その手助けをコンピューターがしてくれる。人間とコンピューターの「正しい」つき合い方は、いつも人間がコンピューターを使いこなすことにあると思います。

このGAME BASIC は、コンピューターを使いこなすための言語としては、非常にわかりやすく、そしてコンパクトにまとまったものです。コンピューターは初めて、という人にも、そしてコンピューターは持っているけど使いこなせなかった人にも、入門用としては最適のものとなるでしょう。

# JUMP



とにかく、この本を開いて、最初のページから順番に読んで下さい。キーボードのキーを押して、と書いてあったら押して下さい。何の予備知識もいりません。最後のページまで読み終わった時、あなたは、いつの間にか1本のゲームを作ってしまいます。そして、コンピュータとは、ベーシックとは何か、なんとなくわかるようになります。それが「ホップ」編です。

「ホップ編」で学んだバラバラの知識が、それぞれ、GAME BASIC でどういう位置にあり、コンピューターとは何かを系統立って説明するのが「ステップ」編です。

「ジャンプ」編では、これまで得た知識を、どのように使って、より高度なものにチャレンジして行くかを説明しています。そして、ファミコンからパソコンへの橋渡しもつとめます。

小学生から大人の方まで、ベーシックの入門書としては今までにない、最初のページから順番に読んで行くだけで理解していただけるものとするように、このような三部編という構成になりました。単にファミコンのファミリーベーシックだけでなく、パソコンのベーシックについてもいろいろな所で触れていますので、いつかあなたがパソコンを手にした時にもお役に立つと信じています。

# もくじ

# FAMILY BASIC HOP




●制作・企画　地球防衛軍　谷岡康則・岩田裕・鈴木康夫・真藤美紀子
簑藤桂・北村順・初瀬龍・美崎靖之
●編集・企画　リポート社

# 作戦指令002
# BASICでの計算は?



# 作戦指令003
# プログラムに挑戦だ!

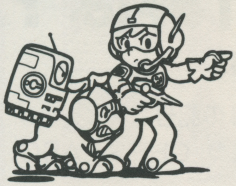


## 巻末資料




●カバー・本文イラスト　高橋裕之（ファクトリー）

# 000 作戦準備指令 接続からはじめよう



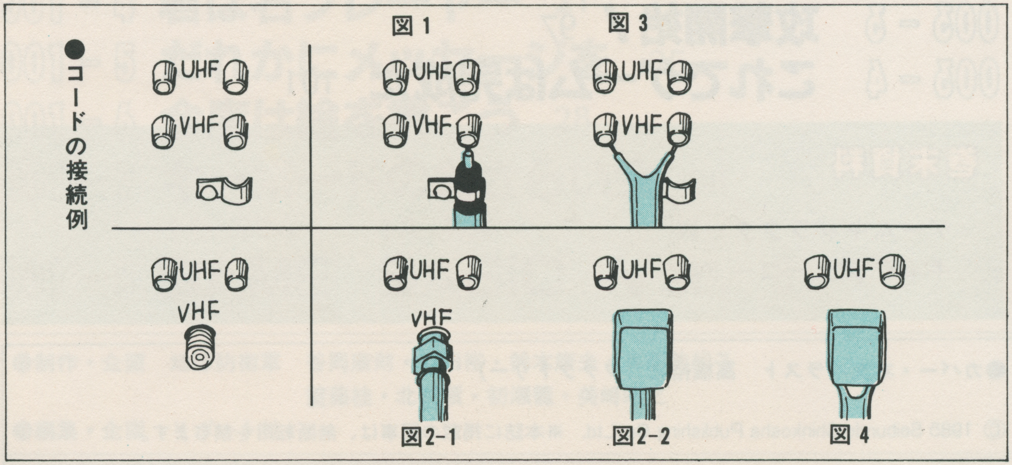
君のファミコンとテレビはもう接続されているだろうか。もしつながっていたら、ただちに作戦指令001を実行してほしい。

新しく接続する君、できたら自分の手でやってみたいね。接続法は、ファミコン本体に付いている「取扱説明書」にくわしく書いてある。テレビとアンテナを結ぶコード（ケーブル）が、どんな形をしているかによって、接続は違うから、よく確かめようね。

テレビとの接続のしかたは、**図1**から**図4**までの方法があるけど、**図2－1**のような場合は、ケーブルを一度切断して芯を出して、ファミコンの RF-SWITCH に接続して、RF-SWITCH からテレビへのケーブルの先には、コネクターを付けなければならない（たいていの電機屋さんに売っている）。芯の出し方は、やはり取扱説明書の17ページにのっている。

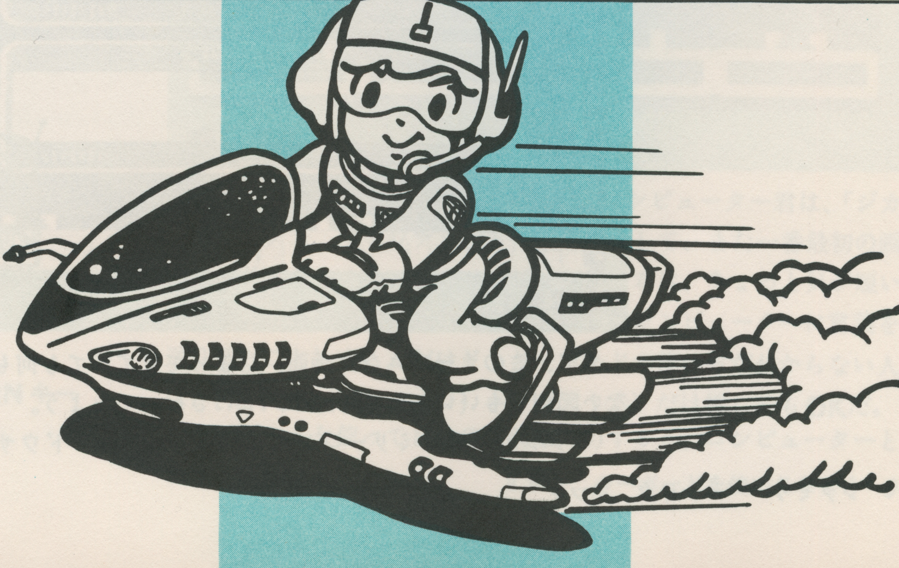
次にキーボードから出ているコードをファミコン本体前面にある差しこみ（エクスパンド・コネクター）に入れて、本体上面のカセット差しこみ口にベーシックカセットを入れれば、準備OK！

これでいよいよベーシック・ワールドへの旅立ちが始まるぞ。

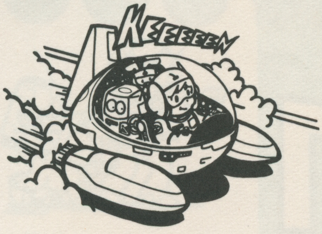


# 作戦指令 001

# キーボードを使いこなそう



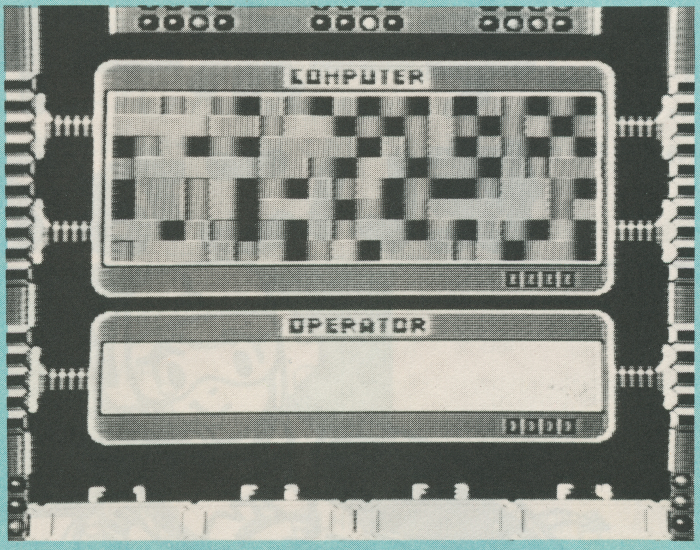
# 001-1



# スイッチを入れたら

すべての接続はうまくいったかな。ファミリー・ベーシックのカセットをセットしたら、いよいよ電源オン！

ピッポッププッという電子音といっしょに、コンピューターらしい画面がテレビ上に見えるはずだ。(**写真１**)。もしも、何もうつっていないという人がいたら残念。もう一度、接続のやり方をよく見て、やり直すこと。電源はコンセントにちゃんと入ってるかな。



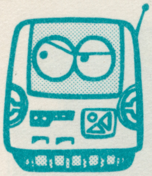

**●写真１**

さて、このオープニング画面、だまって見ていても何もはじまらない。何でもいいから、キーボードにさわってみよう。

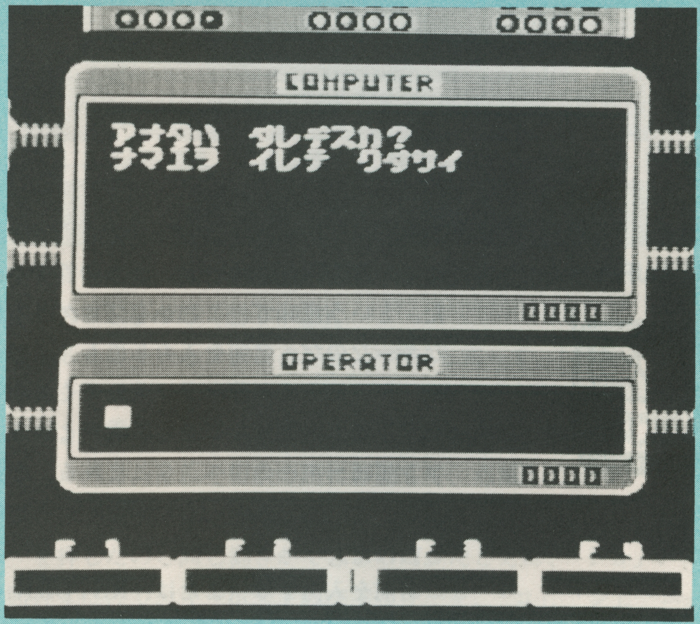
「ワタシハ　ファミリーコンピューター　デス　ドウサヲ　カイ

シ　シマス　アナタハ　ダレデスカ？　ナマエヲ　イレテ　クダサイ」と上の画面に表示されて、下の画面の左がわに、白い四角いものがついたり消えたりしている（**写真2**）。

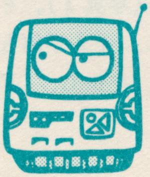
**カーソル**



この四角いものは**カーソル**と言って、君が何かを入力（キーボードを使ってコンピューターに命令したり、答えたりすること）をする時には必ず出てくる大事なしるしなんだ。このカーソルには、四角形以外に、アンダーラインや、羽根ペンといろいろあるけれど、役割はみんな同じだ。




●写真2



さて、このまま何もしないでいると、コンピューター君は、「ジカンギレ　デス　サヨウナラ」と作業を中止して、また一番最初の画面にもどってしまう。このコンピューター君は、ちょっと気が短い。

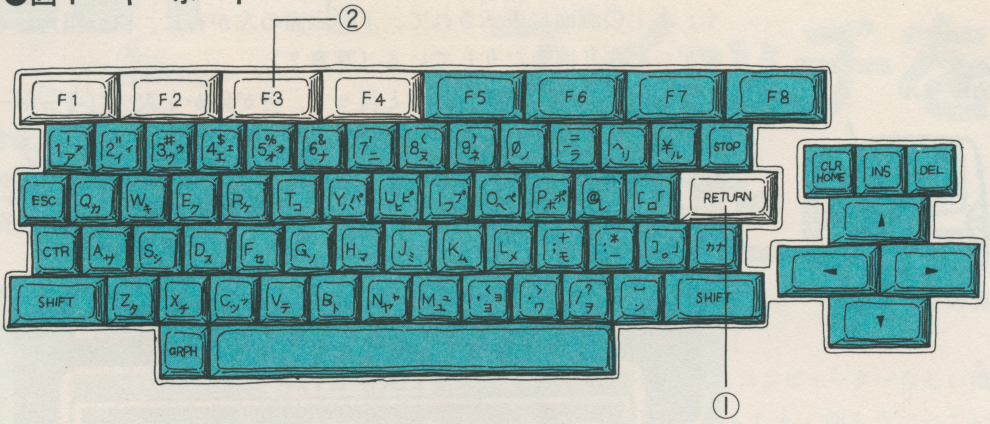
**入力**

**RETURNキー**

本当は、君の名前を**入力**（キーを押してコンピューターに言葉を送り込むこと）するんだけど、キーボードの使い方がわからない人は、**RETURN（リターン）キー**（図1①）だけ押しても**OK**だ。

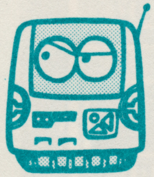
キーボードを使って自分の名前を入力すると、コンピューターと

**●図１　キーボード**




話しができるから気分がいいゾ。名前は18文字まで打ち込める。名前を打ちこんだら、そのあとに必ずRETURN キーを押すこと。

**RETURNキー**

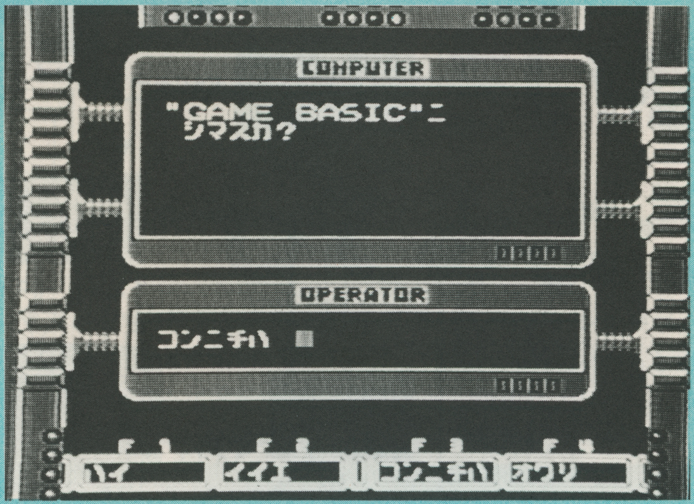


このRETURNキーは、一つの答えや、一つの命令が終わりました、という意味で、これからプログラムを作る時には必ず使う大事なキーなのだ。名前を入力しても、このキーを押さないと、コンピューターはその名前を受け入れてくれない。キーボードから入力した文字は、一度テレビ画面に置いといて、RETURNキーを押すと、コンピューターの中へ送りこまれる、と考えるといい。名前を入れずに、RETURNキーだけを押すと、「名前はないよ」という意味になるわけだ。

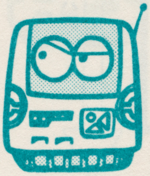
名前を入力する（または、RETURN キーだけを押す）と画面には、「"GAME BASIC" ニシマスカ？」と表示される**（写真３）**。そして、おなじみのカーソルが点滅している。と同時に、今まで何も書かれなかった一番下の四つの窓に字があらわれる。左から、「Ｆ１」、「Ｆ２」、「Ｆ３」、「Ｆ４」と書かれた下の窓だ。

**F1　→　ハイ**
**F2　→　イイエ**
**F3　→　コンニチハ**
**F4　→　オワリ**

カンのいい君はわかったかな。「Ｆ１」から「Ｆ４」は、キーボード




●写真3



**ファンクション・キー**

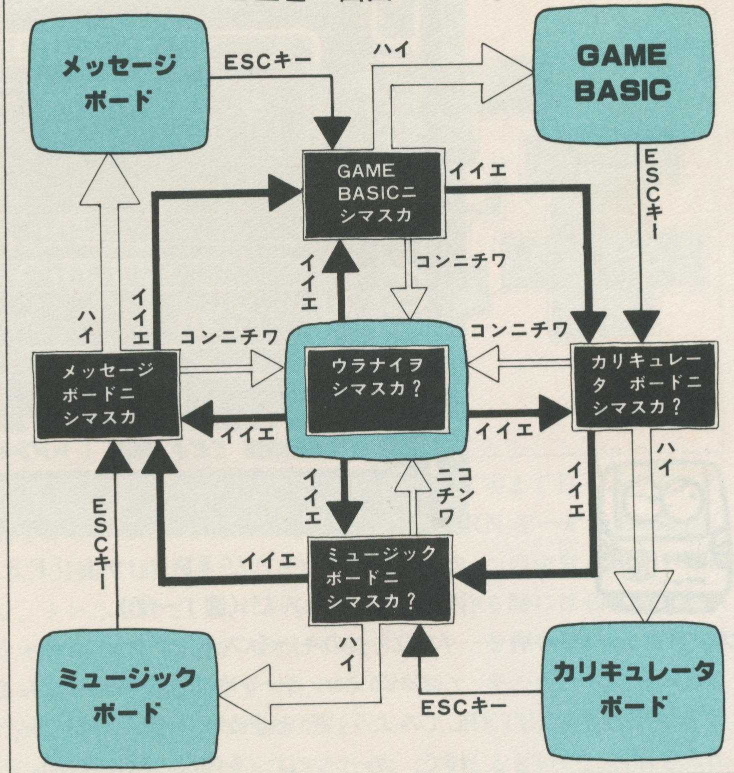
の上段に８つならんだ「Ｆ１」から「Ｆ８」までのキーのうち、左の４つのことなんだ（図１−②）。

この８つのキーを**ファンクション・キー**と言って、「Ｆ１」は「ファンクション・キー・No.１」の略語だ。ためしに、「Ｆ１」キーを押してみよう。今までカーソルが点滅していた所に、「ハイ」と表示されて、カーソルは、その次の所に移動して、点滅している。

このファンクション・キーは、１つのキーを押せば、いくつものキーを押したのと同じ働きができるのだ。たとえば、「Ｆ３」キーを１回押すだけで、キーボードの文字のキーを、「コ」、「ン」、「ニ」、「チ」、「ハ」、と順番に１回ずつ押したのと同じ働きをしてくれる。この場面ではコンピューターが自動的に「Ｆ１」は「ハイ」というように決めてくれているけれど、ゲーム・ベーシックでは、君がどのファンクション・キーに、どんなことばを入れるかを自分で決めることができる便利な機能を持っているんだ。

さて、次のページを見てみよう。このゲーム・・ベーシックVer.2には、計算や伝言板など、いろんなことができるようになっている。

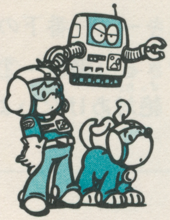
●図２　画面の移り方



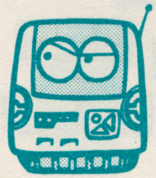

はじめに、「GAME BASIC ニシマスカ」と表示される。「イイエ」と答えると、図のように右まわりに次々とその仕事をしますか、と聞いてくる。そこで、「ハイ」と答えると、その仕事の画面に移っていく。「GAME BASIC」以外は、その仕事を終わりたい時にESC（エスケープ）キーを押すと、矢印の流れにそって、次の仕事の画面に移るようになっている。また、どの画面からでも、「Ｆ３」＝「コンニチワ」キーを押すと、コンピューターが占いをしてくれる。占いからもとの画面にもどるには、「もう一度やりますか」という問いかけに「Ｆ２」＝「イイエ」と答えればよい。

「Ｆ１」から「Ｆ４」までのキーを押して、文字がテレビ画面に表示されたら、次にその文字をコンピューターに送りこむためのRETURNキーを押すことを忘れないこと。

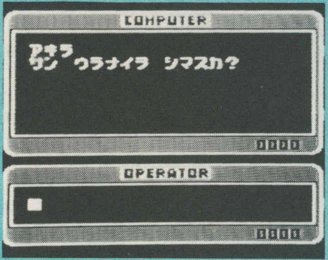
# 001-2 占いをしてみよう



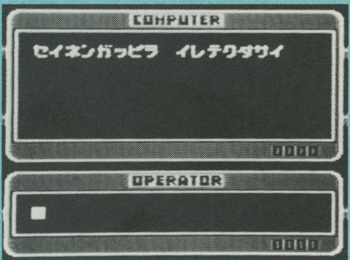
コンニチワ



「F３」キーを押して、つまりコンピューターに「**コンニチワ**」とあいさつをすると、「私は占いができます。占いましょうか」と言ってくれる（**写真４**）。「F１」キーを押して、画面に「ハイ」と表示されたあと、RETURN キーを押すと占いが始まる。




●写真４




●写真５

「セイネンガッピハ」と聞いてくるので（**写真５**）、君は占いたい人の生年月日を入力すればよい。たとえば、君が、昭和48年５月５日生まれだとしたら、

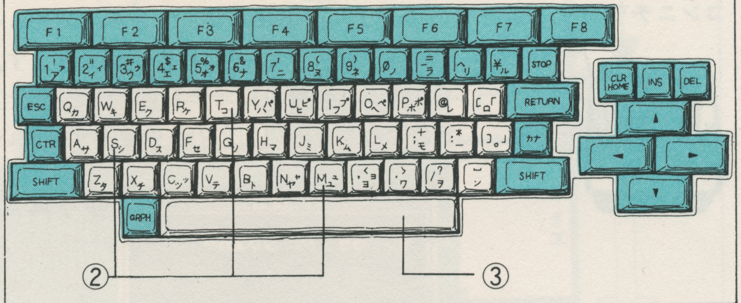
「S」、「４」、「８」、「／」、「５」、「／」、「５」

と押したあと、RETURN キーを押せば、君の生年月日が送りこまれる。TVの画面で「S４８／５／５」となるわけだ。

「S」は「昭和」の略だから、もし大正生れの人なら「T」、明治生まれるおじいさんなら「M」となるわけだ（**図３－②**)。「／」は、年月日のくぎりに入れる記号で、かわりにスペースキー（一番下の横に長いキー）や、数字以外の文字を使うこともできる（**図３－③**)。

「S」または「T」や「M」と、次の数字の間は、絶対あけないようにすること。

**●図３　キーボード**

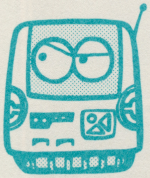



昭和や大正などの年号のかわりに、西暦を使うこともできる。昭和48年は1973年だから、

「1」、「9」、「7」、「3」、「／」、「5」、「／」、「5」

と打ちこんでもOKだ。

次に「占いたい日を入力してください」と聞いてくる。前と同じやり方で、占ってほしい日を入力してみよう（**写真６**)。しばらく考えたあと、ファンファーレとともに占いの結果を表示してくれるはずだ（**写真７**)。



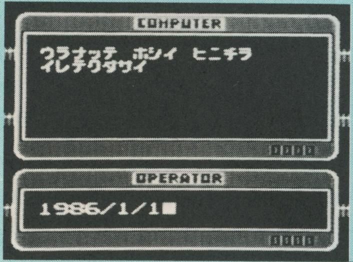
**セレクト画面**

さて、表示が終わると、「続けますか」と表示される（**写真８**)。「ハイ」と答えると、次に占ってほしい日づけの入力を聞いてくる（**写真６**)。「イイエ」と答えると、もとの**セレクト画面**（12ページ）にもどる。別の人（誕生日のちがう人）を占いたい時には、一度セレクト画面にもどって、もう一度あいさつをして、占い画面を呼び出してください。

ところで、「ハイ」のかわりに「YES」と打ちこんでも、「リカイデキマセン」と言われてしまう。けれども、「コンニチハ」のかわりに「HELLO」や「オハヨウ」、「コンバンハ」を使っても、ちゃんと

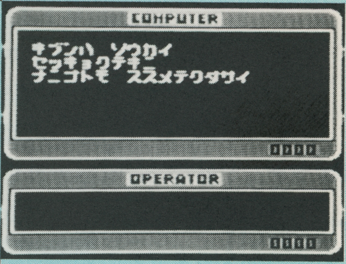
理解してくれることも覚えておこう

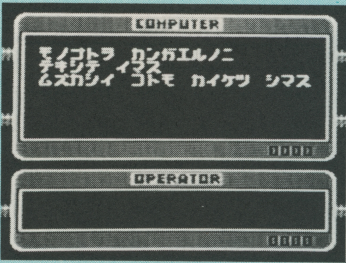
誕生日、占う日は1968年（明治元年）から2924年（昭和999年）までしかできない。ちょっとふざけて君の生まれる前や、100才を越えるおじいさんになった時の占いをしてみたらどうなるだろうか。ためしにやってみたらどうかな。




●写真6

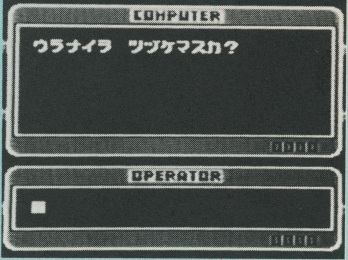
●写真7



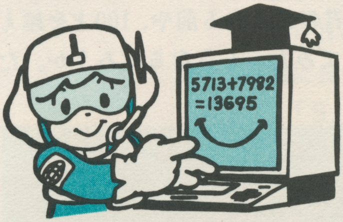




●写真8

●写真9



# 001-3 カリキュレータボード 計算はお手のもの



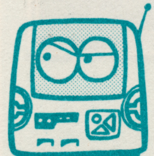

コンピューターにとって、計算は大得意。このカリキュレータボードは、計算専門の黒板というわけなんだ。

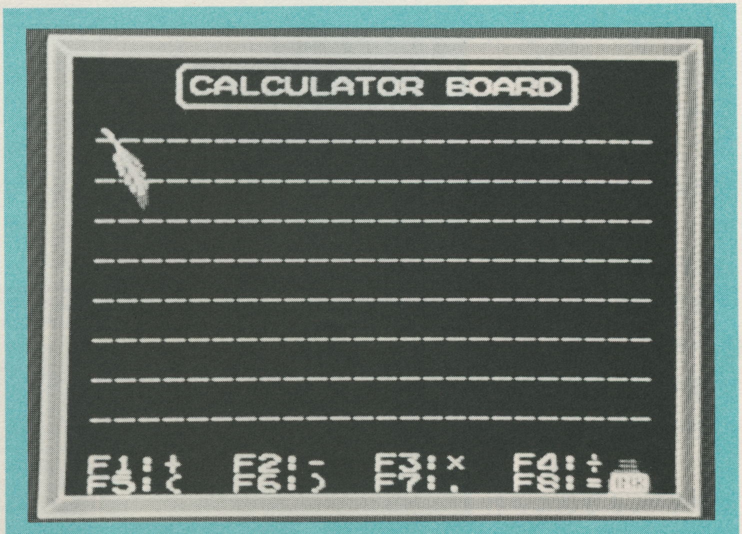
カリキュレータボードに画面が切りかわると、カーソルは白い四角形ではなくなり、羽根ペンに変わる。同時に**ファンクションキー**（「F1」から「F8」までの8つのキーだったね）の中身も、「ハイ」とか「イイエ」ではなくなって、計算に使う8つの記号に変わる(**写真10**)。どのファンクションキーに、どの記号が入っているかは、画面一番下の2段にいつも表示されている。

**ファンクションキー**

それでは、ひとつ計算してみよう。**数字キー**（**図4－①**）と、ファンクションキー（**図4－②**）を使って、

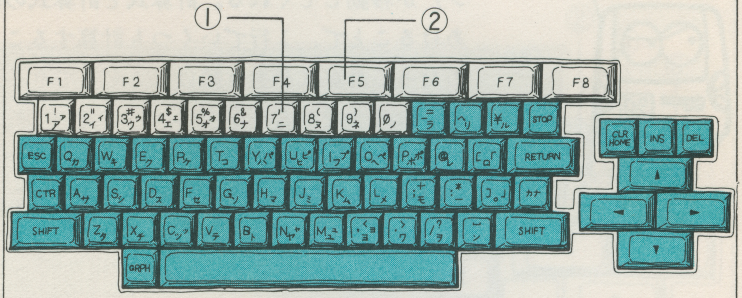
**数字キー**





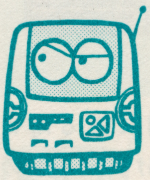

●写真10

●図4　キーボード



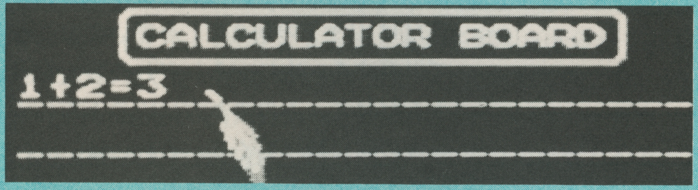

「1」、「F2」、「2」、「F8」
と押してみよう。

画面にはちゃんと、
「1＋2＝3」
と計算結果が表示されて、カーソル（羽根ペン）が答えのあとに1字あけた所に表示されている（**写真11**）。



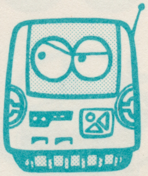
**オーバーフロー**

さあ、どんどん計算してみよう。「F8」（＝）を押すまでは答えは出ないから、長い式や、いくつもの数を続けて足したり引いたりできるわけだ。1行に24文字、たてに8行まで続けた式でもできるはずだよ。ただし、答えや、途中の計算結果が99999999より大きいか、−99999999より小さいと、**オーバーフロー**（計算できる範囲を越えた）エラーとなる。たとえば、99999999＋1−1は、答えは99999999で計算の範囲内だけど、途中で（＋1の所で）範囲を越えてしまうのでエラーになってしまうわけだ。



●写真11

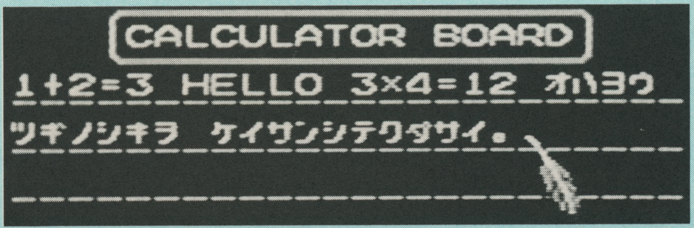
**改行**



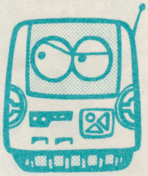
RETURN キーを押すと、改行といって次の行へカーソル(羽根ペン)が移動してくれる。計算式と計算式の間に空白(スペース)をあけることで、1行でいくつも計算することができるし、何行にもわたって計算式を書き続けることもできる。でも、8行目から次の行へは残念ながら続かない。ここまでで終わりなんだ(答えに限って8行目におさまらない時には9行目まで書かれる)。

計算式との間を1字あければ、カナやアルファベットでメッセージを残すこともできる。あとで出てくるカナの入力のやり方を覚えたら、試してみよう。(**写真12**)。

カッコの計算は、残念ながら1重カッコしか使えないし、1つの式の中ではカッコが10組までしか使えない。だから、「3×(2+1+2×(4+3))」なんて式もだめなんだ。




**●写真12**

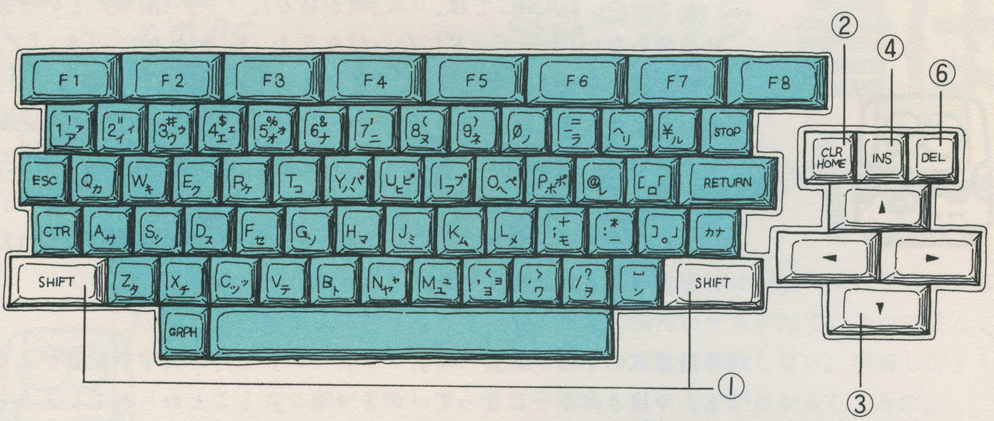


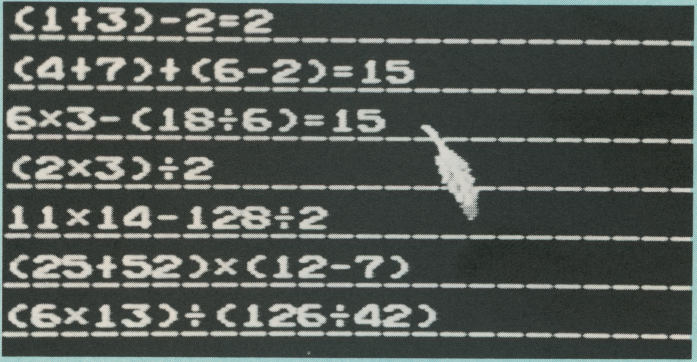
**SIFTTキー**
**CLR・HOMEキー**

画面がいっぱいになって、カーソルが8行目にある時にRETURN キーを押すと、カーソルは1行目にもどるけど、1行目の計算式はそのまま残っている。この計算式の上に、ちがう計算式を重ね書きすることもできる。また、画面全体を一度消したい時には、**SHIFT**(シフト)キー(**図5—①**)を押しながら、**CLR・HOME**(クリア・ホームキー(**図5—②**)を押すことでできる。

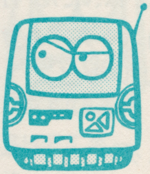
ここで、計算の練習をする方法を教えよう。はじめに各行に計算式だけ(「=」の前まで)を書いておこう。そして、それぞれを自分で計算する。次に、それぞれの答えをコンピューターに出してもらうんだけど、そのやり方は次のようにする。カーソルを各式の一番最後の所に持ってくる。これには、カーソルキー(**図5—③**)を使ってカーソルを移動させればよいのだ。移動できたら、「F8」を押すと、その式の答えが表示されるというわけ。テストの練習には、

**●図5　キーボード**




**●写真13**



**INSキー**
**DELキー**
**エディット**

もってこいだ。(**写真13**)

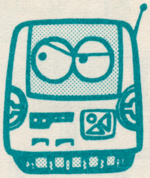
また、すでに書かれた数式をカーソルキーと **INS**(**インサート**)**キー**(**図5ー④**)と **DEL**(**デリート**)**キー**(**図5ー⑥**)を使って修正する(**エディット**という)こともできる。エディットのやり方(69ページにくわしく書いてある)を覚えたら、ためしてみよう。

〔注〕 ただ、注意しなければいけないのは、このINSとDELは、もとの数式が何行にわたっていても、１行分の中でしか働かないことだ。マニュアル(説明書)には書かれていないけれど、何行にもわたる数式では、エディットはできない(69ページを

見てください)。

もう一つ、BASICでは、「×」(かける)、「÷」(割る)という記号は使わない(キーボードにないだろう!　)。かわりに、「＊」「／」という記号を使う。たとえば、「4×3÷2」ならば「4＊3／2」というように書く。ところが、同じファミコンでも、このカリキュレータボードに限って、「×」、「÷」を使わなければいけないんだ。キーボード上の「＊」や「／」を使うとエラーになってしまう。「×」、「÷」は、ファンクションキーに登録してあるので、それだけを使うことになっている。

「＊」「／」



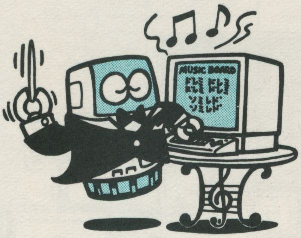
計算式の中に「×」、「÷」、「＋」、「－」以外の文字や記号が入ると、やはりエラーになって「デキマセン」とことわられてしまうし、式の途中に空白があると、その空白のあとの部分しか計算してくれない。だから空白の次の部分が計算式になっていない(たとえば、「3＋2　＝」など)とエラーになる。また、メッセージ(文章)を行の最後まで書いて、次の行で計算式を続けて書くと、メッセージの文字と計算式が続いているものとされて、エラーになってしまう。こういう時は、メッセージを書き終わったら、スペースキーを1回押して空白を作ればOK。

普通は、RETURNキーは、1つの式や文字を送り出す、「切れ目」を表わすものだけど、このカリキュレータボードの中では、ただ「行をかえる」意味しかなく、かわりに「空白＝スペース」(1文字あける)が、「切れ目」の役割をはたしていると考えればよいのだ。

カリキュレータ画面を終わりたい時は、ESC(エスケープ)キーを押すことで、次の機能を選べる(12ページ　図2)。また、電源を切るか、GAME BASICを始めるか、本体のRESET(リセット)ボタンを押さない限り、書きかけの画面はコンピューターに保存されている。

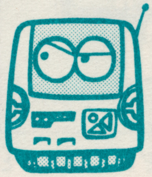
# 001-4

## ミュージックボード 君は名プレーヤー



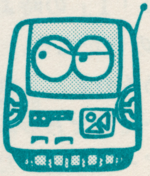

ミュージックボード画面でも、カーソルは羽根ペンになっている。画面は4行になっていて、左上に四角いマスがたてに3つに並んでいる。このマスは、実際の演奏中にしか移動しない。演奏しだすと、右へ動いて行って、どこを演奏しているかがわかるしくみだ。

この演奏マスは3段になっていて、これは、上から高音・中音・低音の3つの音域に分かれているのだ。さっそく音を出してみよう。



ファンクションキー

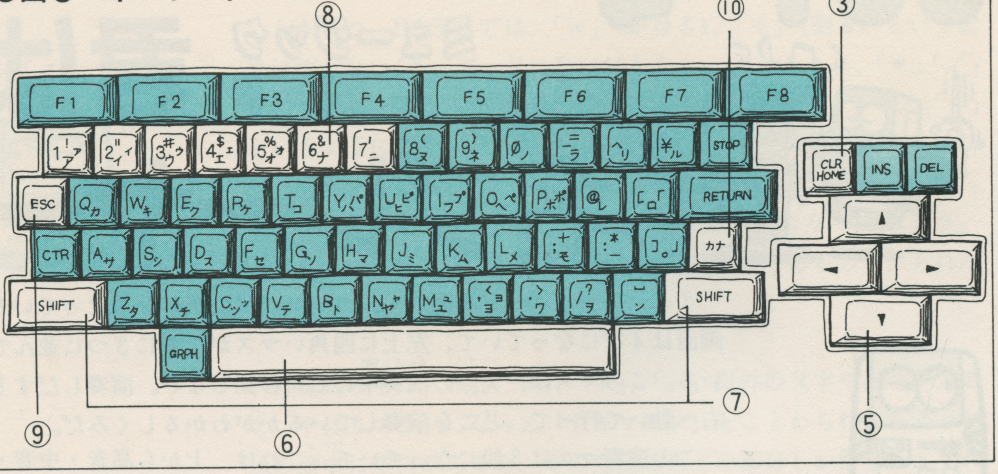
まず、音の入力は**ファンクションキー**で行なう。ファンクションキー（図6－①）は、「F１」から「F７」までが、それぞれ「ドレミファソラシ」の7音になっている。「F１」を押してみよう。「ド」の音がポンと出て左上に「ド」と書かれたはずだ。そして、カーソルは1文字に移動する。「F１」から「F７」をでたらめに押してみよう。それぞれの音を出しながら、画面に「ドレミファソラシ」のどれかの文字が書かれて、カーソルはどんどん右に移動して行く。1行に24文字で、4段96音まで書ける。ファンクションキーのかわりに、スペースキー（図6－⑥）を押すと、1文字分、何も書かない（空白）で、カーソルだけが右へ移動するだけだ。



SHIFTキー

今度は、**SHIFTキー**（図6－⑦）を押しながら「F１」を押してみよう。カーソルのあった所には「C」と書かれて、「ド」と「レ」の中間の音が出たね。つまりSHIFTキーを押しながらファンクションキーを押すと、そのファンクションキーに登録されている音より半音高い音が出るしくみなのだ。普通は、この音は「ド#」とか、「C#」と書くけど、このミュージックボードでは、1文字で1つの音を表わすため、こんな書き方をしている。音楽では「ド」も「C」も同じ音だから間違いやすいけど、アルファベットで表わされた音は、半音だけ高い音、ピアノでいえば黒鍵にあたると覚えておこう。

**●図6　キーボード**

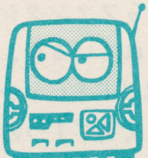



もちろん「ミ#」(「E」)と「シ#」(「B」)は、それぞれ「ファ」と1オクターブ高い「ド」と同じ音になるのは知っているよね。

さて、次はファンクションキーの下にある数字キーの「1」から「7」(図6－⑧)を押してみよう。「1」を押すと、さっきの「F1」より1オクターブ低い「ド」の音が出て、画面には「1」と書かれている。今度はカナキー(図6－⑩)を1回押して「1」を押してもらおう。画面には「ア」と書かれて、さっきの「1」の時より半音高く「C」(「ド」の#だね)の1オクターブ下の音が出る。

図7を見てください。上段Aとか下段Bと書いてあるけど、これはファンクションや、SHIFT＋ファンクションで書いた時をA形式(ド～シ、C～Bと表示される)、数字の1～7や、ア～ニで書いた時をB形式とした時、どの音が出るかを示したものなんだ。

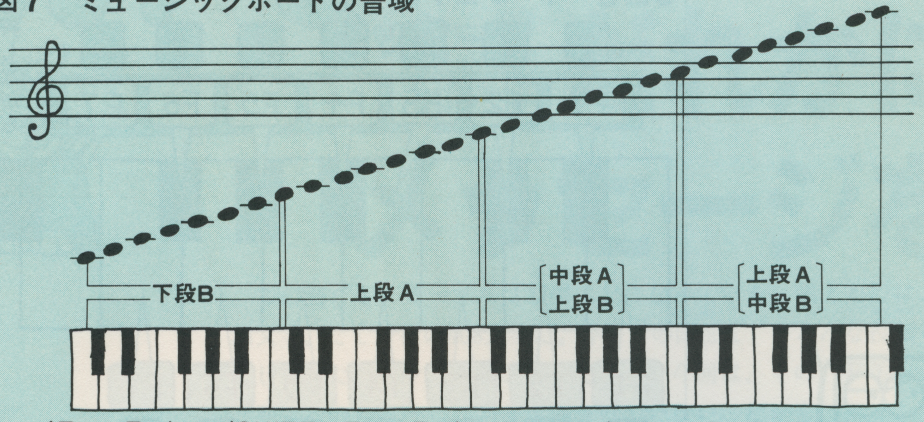
この上段とか中段、下段というのは、はじめに言った演奏マスの3段の上中下という意味だ。さて、ここで整理をしておこう。演奏できる音域は、4オクターブになるわけだけれども、この内、最低音(一番左)の1オクターブは、下段にB形式(1～7やア～ニ)で書く。一番右の最高音の1オクターブは、上段のA形式か、中段のB形式で書くことで音が出るというわけだ。



**カーソルキー**

中段と下段に書き込む方法はというと、**カーソルキー(図6－⑤)**の4つのキーを使えばできる。下向きのキーボード「▼」を押せば、

●図7　ミュージックボードの音域

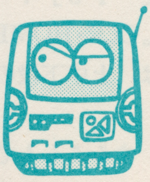



A＝（F１～F７）か（SHIFT＋F１～F７）を使った時（ド～シ、C～Bで表示）
B＝（１～７）か（ア～ニ）を使った時（１～７、ア～ニで表示）

カーソルが一つ下に動くのだ。ここでためしに「F１」を押すと、「ド」と表示されて、さっきの上段の「１」の音と同じ音が出るはずだ。

ただし、このカーソル、ちょっとくせがある。たとえば、今１行目の下段にカーソルがあるとする。カーソルキーの下向きを１回押すと、１行目の下段と２行目の上段の間に止まってしまう。この位置では、どんなキーを押しても音はでないので、もう一度下向きのカーソルキーを押して、２行目の上段へ持っていかなくてはならない。

ところで、今「１」を押すと、どんな文字が出るかな。「ア」となるはずだ。ここで、もう一度カナキーを押してから、「１」を押すとどうかな。今度は「１」と出たね。「C」や「D」を出すには、いつもSHIFTキーを押しながら、ファンクションキーを押さなければならなかったけど、このカナキーは、一度押すと、ずっとカナになる。もとの数字を出すには、もう一度カナキーを押さなければならない。何回もためしてごらん。

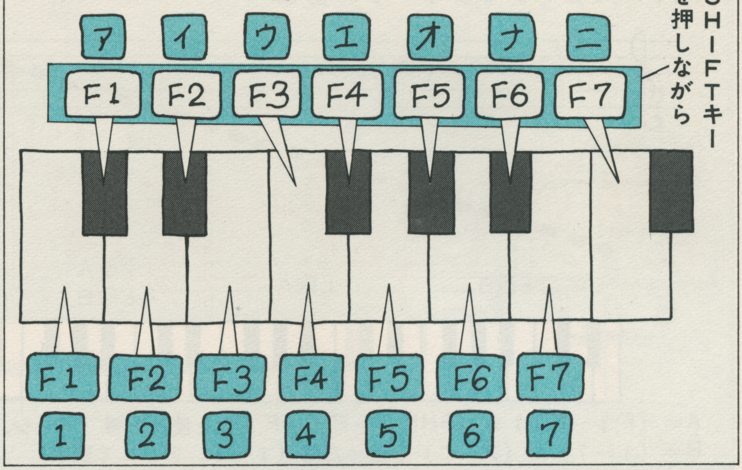


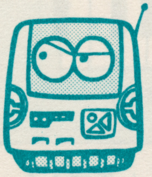
**演奏**

さて、ここで「F８」を押してみよう。ワクが移動しながら、今まで書きこんだ音を順番に演奏してくれる。４行目の最後まで行くと、演奏終わり。途中で演奏をストップしたい時は、演奏中にスペースキー(図６－⑥)を押す。また、演奏中にCLR・HOMEキー(図

**演奏ストップ**

●図8　キーが出す音



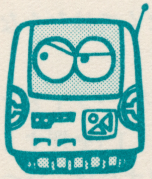



**書き込みモード**

6－③）を押すと、演奏を中止して、**書き込みモード**にもどる。でも、これをSHIFTキーを押しながらCLR・HOMEキーを押すと、前にも言った通り、画面に書いた音符がみんな消えてしまうから注意すること。

**スピード変更**

演奏中にカーソルキーの「◀」、「▶」を押すと、演奏のスピードが変わる。これで、スローテンポにもアップテンポにもできるわけだ。また、スペースキーで演奏を中止した状態で「◀」、「▶」を押すと、今止まっている所から、前後へ1回押すごとに1音ずつ演奏してくれる。「F８」キーを押すと、そこからあとを続けて演奏する。そして、もう一つの機能として、こんなことができる。カーソルキーで、カーソルを好きな所に移動してSHIFTキーを押しながら「F８」キーを押すと、「$」と書かれる。このあと「F８」キーを押すと、はじめから「$」の所までを何回もくり返す。CLR・HOMEキーでストップさせるまで止まらない（この「$」は、上中下段のどこに書いてもいい）。

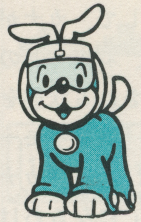


**くり返し演奏**

INS、DELキー（図6－④）でエディット（修正）することもできるけど、カリキュレータボードと同じく、1行中だけだから注意すること！

ミュージックボードをぬけ出すのは、やはりESCキー（図6－⑨）だ。

# 001-5 メッセージボード だれかにメッセージを



この面でも、カーソルは羽根ペンになっている。使えるのは、横24文字、たて17行の408文字だ。今までの画面で、いろんな役割をはたしてきたファンクションキーだけど、今回は、その「Ｆ１」から「Ｆ７」の各キーには何も定義されていない。だから、ファンクションキーを押しても何も出ないし、変化もしない。

メッセージの内容は何を書いてもよいし、RETURN キーや、カーソルキー、CRL・HOME キーの使い方も今までの画面とまったく同じなんだ。

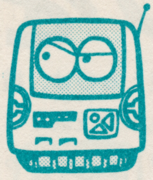



●写真14

**カナキー**

**カナ文字モード**

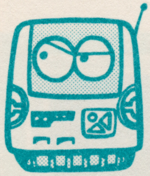
**英数字モード**



この画面では、文章をすらすら書けるように、カナの書き方を覚えよう。**カナキー**（**図9—①**）を1回押すと、カナ文字モードになるというのは覚えているかな。キーボードのキーを押して、カナが画面に表示される状態を「**カナ文字モード**」と言って、羽根ペンの色が緑色になる。アルファベットや数字・記号が表示される状態を「**英数字モード**」と言って羽根ペンの色は赤色になる。「英数字モード」でカナキーを1回押すと、「カナ文字モード」になって、「カナ文字モード」でカナキーを1回押すと「英数字モード」になるわけだ。

さて、「カナ文字モード」でキーを押すと、カナが画面に表示されるけど、小さな「ョ」や「パ」や「バ」はどうやって入力すればよいのだろうか。

**SHIFTキー**

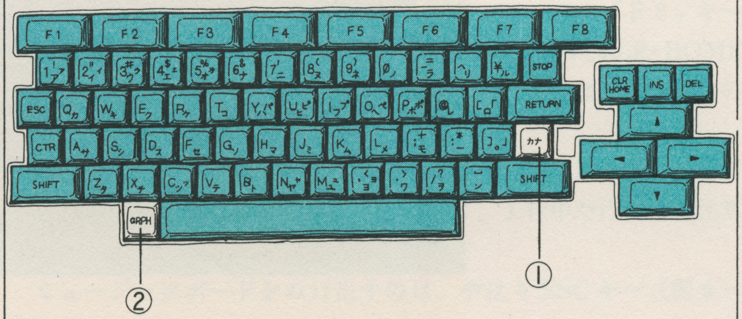


小文字のカタカナは、ア行とヤ行と「ツ」だね。キーボードのア行や、「ツ」の所を見てみよう。大文字の「ア」や「ツ」の右上の「ァ」や「ッ」が書いてある。この右上の小文字を出すには、**SHIFTキー**と「ア」のキーを同時に押す。ちゃんと小文字の「ァ」が出てくる。

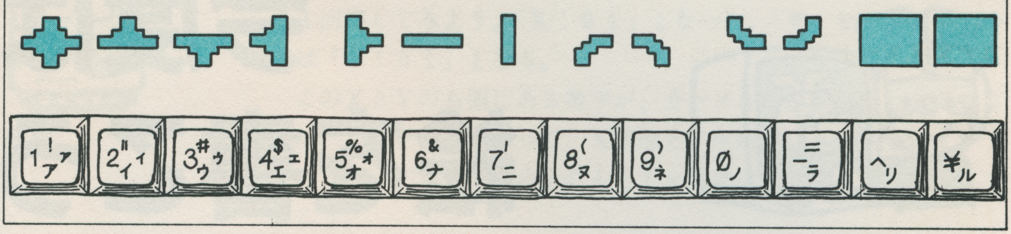
「パピプペポ」も同じこと。「ハヒフヘホ」のそれぞれのキーの右上に「パピプペポ」と書いてあるね。SHIFTキーと「ハ」のキーを同時に押すと「パ」と出る。会話の時のカギカッコもキーボード右に「。」と「□」のキーの右上に書いてあるから、同じようにSHIFTキーを使ってやればよいのだ。

「バ」などの濁音はちょっと違う。キーボードには何もそれらしき

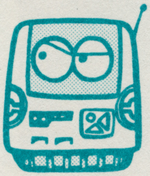
**●図9　キーボード**



●図10　GRPHキーキャラクターを表示



**GRAPHキー**

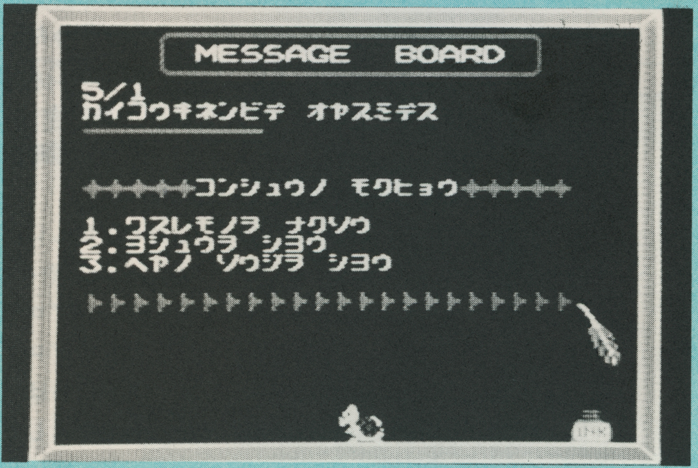


ものはない。これを出すには、左下の **GRPH（グラフィック）キー（図９－②）** を押しながら、「ハ」のキーを押す。これで「バ」と表示される。濁音のあるキー（カ行やタ行など）とGRAPHキーを同時に押せばいいわけだ。「マ″」なんていうのはだめだよ。

今度は、GRPHキーと、文字キーの一番上の段（１ア～¥ル）のキーを同時に押してみよう。ちょっと変わった記号が出てきたでしょう（**図10**）。この記号を使って、画面上にいろいろな線が書けるんだ（**写真15**）。

DEL、INSとカーソルキーを使ってエディットすることもできるけど、注意点は前と同じ。69ページを見てほしい。

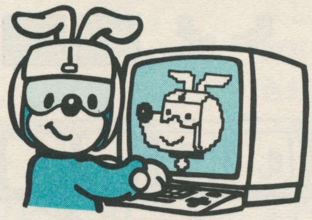
メッセージボードからの脱出は、やはりESCキーを押す。

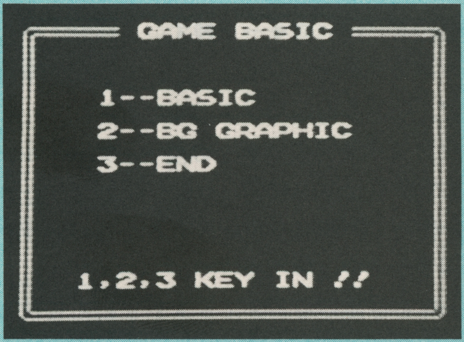



●**写真15**

# 001-6

# 今度は絵を書こう





●写真16



●写真17

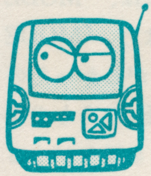
さて、いよいよ「GAME BASIC」に「ハイ」と答える番がきた。この面は、ゲームをやったり、作ったりする所なんだ。

ところが、すぐ「GAME BASIC」にはならない。こんな画面が出てきた（**写真16**）。ここで、数字キーの「3」を押してみよう。ゴメン！　ゴメン！　また一番最初の画面にもどってしまった。やり直し。

さて、ここでは「2」を押してほしい。RETURN キーは押さなくてもよい。「BG GRAPHIC」からやるわけなんだ。

実は、この「GAME BASIC」、プログラムを記憶する場所がとてもせまい。だから、バックの絵をプログラムの中で書いていたりすると、それだけでプログラムがいっぱいになってしまう。そこで、動かない背景の部分だけを別扱いにして作るのが、この「BG GRAPHIC」なのだ。「BG」は、バック・グラウンドの略だ。

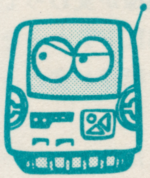
「2」を押すと、写真17のようになる。カーソルは白い四角でもな



く、羽根ペンでもない。四角いワクになって左上にある。左下には「X：００」「Y：００」と書いてある。ためしにカーソルキーの「▶」を１回押してみよう。「X：０１」となった。「▼」を１回押すと今度は「Y：０１」となる。

座標

このXとYの右側にある数字が、カーソルの位置を表わしているのだ。これを**座標**といって、X座標とY座標という。「座標」という考え方は、プログラムの上でもとても重要になるからよく理解しておこう。



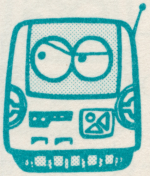
たとえば、君の作ったゲームの主人公がどこにいるかを表わすのに、この座標を使うわけだ。普通のファミコンの場合は、横28、たて24マス目で表わす(**図11**)。でも、このBG画面では、下の２行を使っているから、たては21までしか使えない。上から11番目、左から16番目のマス目にいるとしたら、「X＝10、Y＝15」と表わされる。えー！　「X＝11、Y＝16」じゃないのかって？　コンピュータの場合、ほとんど最初は**０から数える**というのが約束なのだ。たとえば、左から５番目は、「０、１、２、３、４」で、５番目は４だからX＝４になる。

０，１，２，……

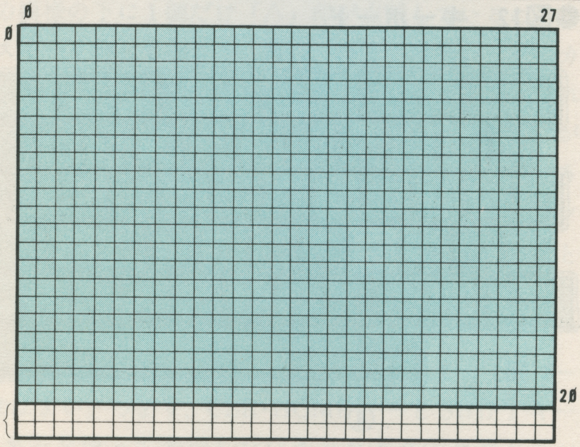
つまり、**一番左上**のマス目は「X＝０、Y＝０」になる。このきまりは、ほとんどのパソコンが取り入れている。(一部のコンピューターは左下すみをX＝０、Y＝０にしているものもある。)

X＝０，Y＝０

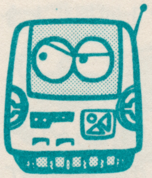
カーソルキーをいろいろ動かしてみよう。右下すみは「X：27」、



●図11
画面の座標



「Y：20」になったよね。さて、その「X：27」の横には、「MODE 0」と書いてある。今度は、RETURN キーを押してみよう。「MODE 1」と変わって、その横に並んだ模様の色が変わったのがわかったのかな。何回も RETURN キーを押してみると、「MODE」のあとの数字が「0」「1」「2」「3」と4通りに変わって、色も4通りがくるくる変わっていくのがわかるでしょう。つまり、同じ模様でも4つの違った色が使えるわけなんだ。この「MODE」の横の数字は**カラー番号**なんだ。



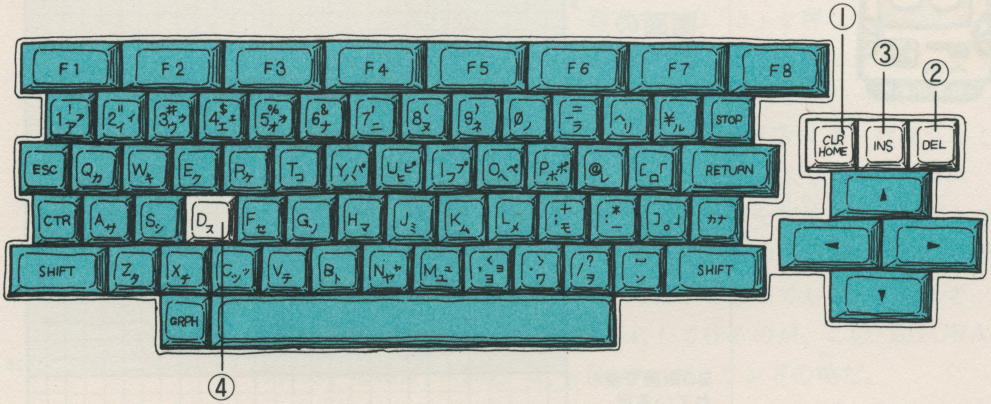
**カラー番号**

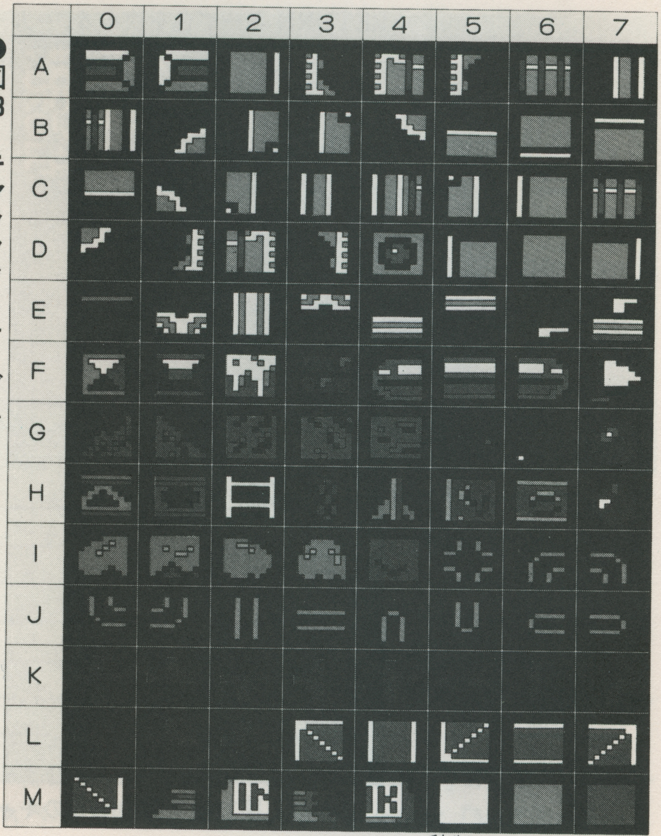
**キャラクター**

さて、この8つ並んだ模様。これを**キャラクター**という。今度は、CLR・HOME キー（**図12—①**）を押してみよう。8つとも模様が変わったでしょう。何回も押してみると、そのたびに模様が変わっていく。13回目からはもう変化しなくなったね。では、SHIFT キーを押しながら、CLR・HOME キーを押してみよう。また模様が変わった。でも、この模様、見たことがあるね。

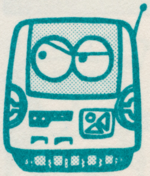
**図13**を見てみよう。一番最初に、この BG 画面にした時、図の一番上に並んだ8つの模様が出ていたはずだ。CLR・HOME キーを1回押すと一つ下の組が表示され、SHIFT キーを押しながら CLR・HOME キーを押すと、一つ上の組が表示される。一番上と一番下の組が表示されている時は、それより上や、それより下は出ないわけだ。そして、その組が全部で12組、つまり96個のキャラクターが書けることになる。

**●図12　キーボード**






●図13 キャラクターテーブルB



セレクト・カーソル

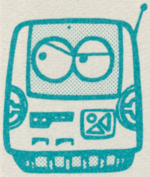
さて、画面の模様の下に1個だけ「∧」印が点滅しているね。ここではスペースキーを押してみよう。あれ、今まで四角いワクのカーソルがあった所に「∧」印の上にあったキャラクターが表示された。この「∧」を**セレクト・カーソル**と呼ぼう。じゃあ今「∧」が点滅している所のキャラクターの右のものを表示したいけど、どうしたらよいだろう。DEL キー（図12—②）を押してみよう。セレクト・カーソルが1個右へ移動したね。ここでスペースキーを押すと…、お見ごと、思った通り、右どなりのキャラクターがカーソルのあった所にあらわれた。

セレクト・カーソルを左に動かすには、今度はINS キー（図12—③）を1回押せばよい。つまり、CLR・HOME キーとSHIFT キーとINS、DEL キーを使って表示したいキャラクターを選んで、

**表示**

RETURNキーで色を決める。一方、カーソルキーで表示したい場所を選ぶ。そして、**スペースキー**をポンと押せば、選んだキャラクターが選んだ場所に表示されるというわけだ。

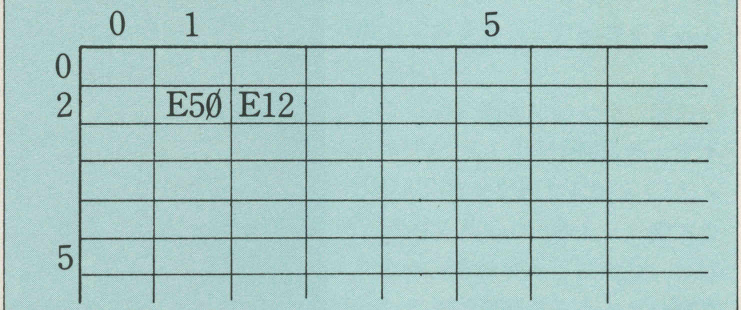
**とり消し**



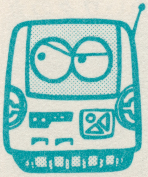
もし、一度書いたキャラクターを消したい時は、カーソルキーでカーソルを消したい場所へ持っていき、**「D」のキー（図12ー④）**を押せば、そこだけ消えてなくなる。

「ファミコン　ソフト集」を見ると、BG画面のデーターがのっている（図14）。これは、たとえばX＝1、Y＝1には「E５０」のキャラクターを入れてくれ、という意味だ(X＝1、Y＝1の所に「E５０」となっているね)。

●図14　BG画面データー



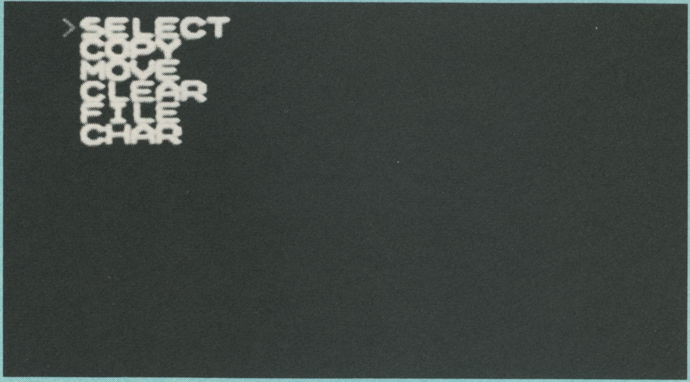

これは、図13のE列の№.５のキャラクターを、カラー０で入力してくれということだ。まず、カーソルをX＝1、Y＝1の位置に合わせて、RETURNキーでカラー（MODEの右横の数字）を０する。次に、CLR･HOMEキーとSHIFTキーで、E列のキャラクターを呼び出して、DELかINSキーで№.５（左から６番目）のキャラクターの下にセレクトカーソルを移動してスペースキーを押すというやり方だ。



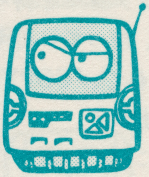
**ESCキー**

ここで**ESCキー**を押してみよう。今までESCキーは元の画面にもどるためのキーだったけど、今度は違う。写真18のような画面が出てくる。

たてに６つの単語と、その左に「>」の印が点滅している。これは、BG画面を書くのに便利ないろいろな機能が選べるようになっているのだ。考えるより実行、さっそくやってみよう。




●写真18

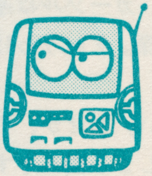


カーソルキーの「▲」と「▼」を押すことで「>」マークが上下するね。この「>」マークを自分が使いたい機能の左に合わせて、スペースキーを押すと、その機能が使えるというわけだ。

**SELECT**

**「SELECT」**の左に「>」を合わせて、スペースキーを押してみよう。あれ、もとの画面だ。さっきの「MODE」の文字の下にも「SELECT」と書いてある。つまり、今まで説明したことは、この「SELECT」機能のことだったんだ。

**COPY**

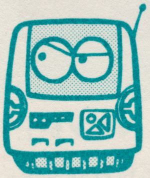


もう一度ESCキーを押して、今度は**「COPY」**の所に「>」を合わせてスペースキーを押してみよう。今度は「MODE」の下に「COPY」と表示されて、右下のキャラクターセットや「∧」がなくなってしまった。

カーソルキーでカーソルを動かしてみよう。何かキャラクターが書かれている場所へカーソルを合わせる。そこでINSキーを１回押して、次にカーソルを動かしてみよう。どうかな。カーソルのワクの中に、今INSキーを押した場所にあったキャラクターと同じものが入っているでしょう。今度は、そのカーソルを何も書かれていない所へ持っていって、DELキーを押してみよう。何も変わらない？　そうまだ変わらない。前のようにカーソルを動かすとどうなるか。カーソルを一つどちらかへ動かしてみよう。はい、今カーソルがあった所に、カーソルのワクの中にあったキャラクターが残ったね。つまり、この「COPY」モードは、INSキーを押した所にあったキ

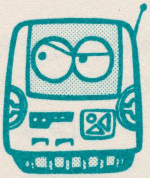
ャラクターを、DEL キーでいくつもコピーできる機能なのでした。
「COPY」モードが「SELECT」モードと違うのは、キャラクターを書いたあと、カーソルが移動しない点だ。ためしに、両方のモードで同じキャラクターをたて一列に書いてごらん。「COPY」モードの方がずっと楽でしょう。

さて、拾ってきた（つまり、カーソルの中にある）キャラクターを捨てて、別のキャラクターをコピーしたい時だ。目的の新しいキャラクターの所にカーソルを合わせて、INS キーを押せば OK。また、カーソルのワクの中に何も書かれていない状態で、キャラクターの所にカーソルを合わせて、DEL キーを押して、そのキャラクターを消すこともできる。逆に、カーソルが何かキャラクターを持っていて、何も書かれていない所で INS キーを押すと、持っていたキャラクターを捨てて、ワクの中はからっぽになる。これで、「COPY」機能の説明は終わり。何度もやって完全に覚えてしまおう。



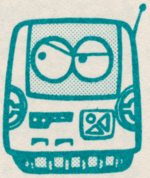
**MOVE**

さて、次は「**MOVE**」機能だ。これも「COPY」に似ている。目的のキャラクターの上側にカーソルを合わせて、INS キーを押すと、今度は、キャラクターがカーソルのワクの中に移動してくる。COPY と違ってキャラクター自身が移るわけだから、カーソルを動かすと、もとの位置には何も残っていない。運びたい場所まで持っていって、DEL を押すと、その場所にキャラクターが置かれて、カーソルのワク内には何もなくなるのだ。この場合は、カーソルは運ぶ役で、INS キーは拾う作業、DEL キーは捨てる作業。というわけで「MOVE」機能はわかったかな。



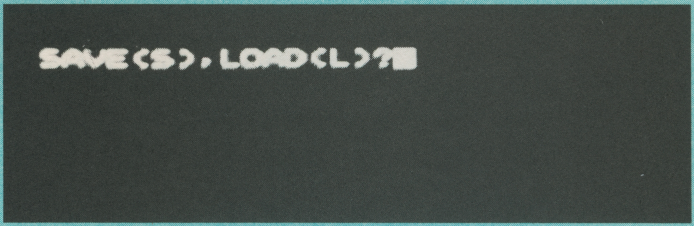
**CLEAR**

続いて「**CLEAR**」機能。こいつは気をつけてほしい。なぜか、やってみてごらん。画面を全部きれいさっぱり消してしまう働きをするからだ。せっかく入力したグラフィックが全部消えてしまい、もとにもどす方法はないからなんだ。

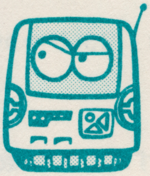


**CHAR**

ところで、この BG 画面では、**図13**のキャラクターしか書けないのだろうか。数字やカナも書きたいんだけど、という人のために、ちゃんと用意がしてある。これが「**CHAR**」という機能だ。ここに「>」を合わせてスペースキーを押すと、カーソルが白い四角に変わる。カナキーを押すと、今度はアンダーラインのカーソルだ。つまり、ここではメッセージボードで使ったのと同じキーの使い方で、



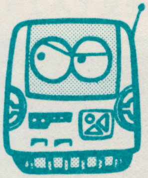

●**写真19**



FILE

英数字やカナが打ち込めるのだ。ただし、INS や DEL キーを使ったエディットや、CLR・HOME キーでカーソルを左上へ持っていったり、全部の画面を消すという機能は使えないよ。

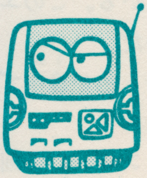
さて最後に残った機能、「**FILE**」に「>」を合わせてスペースキーを押そう。今まで書いた画面が全部消えて、**写真19**のようになる。



SAVE
LOAD

コンピューターに書き込む所を RAM というのだけれど、これは電源を切ると書き込んだものが全部忘れられて、なくなってしまう（ステップ編作戦資料002)。そこで、今まで書き込んだプログラムや、絵のデータを何か他のものに保存する必要があるんだ。保存さえしておけば、また次に読み出して使えるからだ。ファミコンでは、保存はカセットテープに録音することで行なっている。この保存を「**SAVE**」（セーブ）と言い、また、保存したテープから読み出すことを「**LOAD**」（**ロード**）という。これらを行なうのが「FILE」機能なのだ。

保存したい時は「S」を押し、読み出す時は「L」を押す。保存するといっても、せっかく書いた絵が見えないじゃないかと心配する人。君が「CLEAR」機能で全部消していない限り、見えないだけで全部残っている。ご心配なく。ただし、「LOAD」読み込みを選ぶ時は気をつけなくてはいけないことがある。保存していたデータを全部画面に書き込むために、今までコンピューターに書いていたデータはなくなって、テープのデーターに変ってしまう。一度 LOAD が終わってしまうと、もうもとにもどす方法はないのだ。何か絵を書いて、それを残しておきたい時は、必ず SAVE をすること。



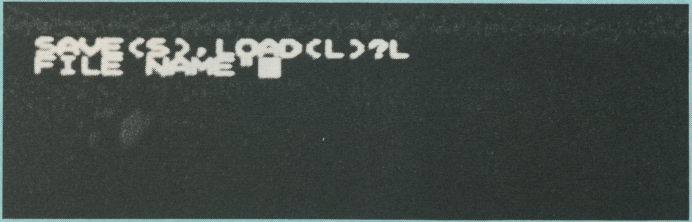
FILE NAME

「S」か「L」を押すと、「**FILE NAME**」（**ファイル・ネーム**）と聞いてくる。(**写真20**)。これは、カセットテープに何種類もデータ

やプログラムを入れる時に、他のものと区別するために、君がつけた名前のことだ。SAVE する時に16文字以内で入力し、RETURN キーを押すと、SAVE が始まる。LOAD の時も、同じファイル・ネームを打ち込んで RETURN キーを押すと LOAD が始まる。もし、SAVE した時のファイル・ネームを忘れたら、ただ RETURN キーを押すだけでも OK だ。一番最初にあるものを読みこんでくれる。（くわしくは、作戦指令002-4を読んでね。）

この BG 画面の SAVE、LOAD には、それぞれ約１分30秒かかる。

ここまでで「BG GRAPHIC」の機能はわかったはずだ。カナ・数字・記号、そして96種類のキャラクター、これは４色だから96×４＝348種類。これらを使って絵が書けるわけだ。でもちょっと待ってほしい。機械のつごうで、少し問題点があるんだ。図15を見てみよう。



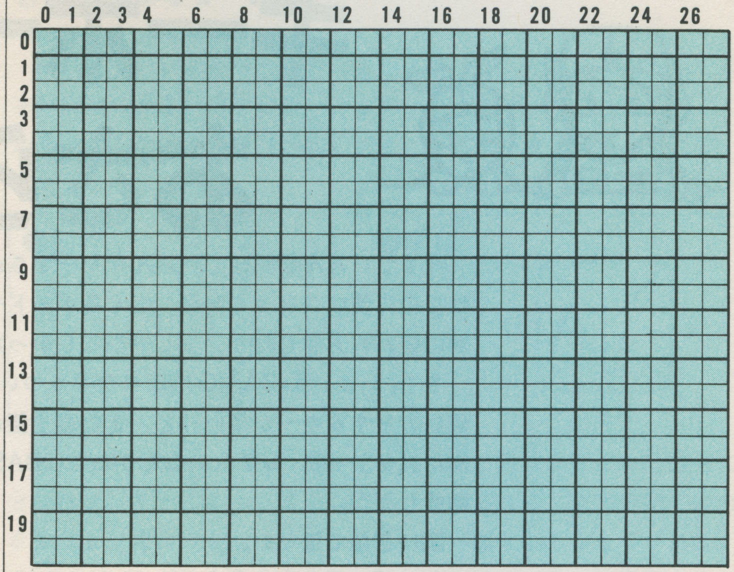

●写真20

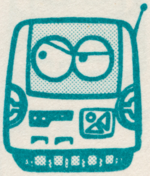
太い線で囲まれたエリアの中では、１つの色しか使えないんだ。たとえば、X＝２、Y＝１へカラー０（ブルー）でキャラクターを書いたあと、X＝３、Y＝２へカラー１（オレンジ）のキャラクターを書いてみよう。X＝２、Y＝１に書かれたキャラクターの形はもとのままだけど、色がカラー１に変わってしまっている。このエリアの中では最後に使われた色だけになってしまうのだ。

だから、最初に絵の組み立てをよく考えて書かないと色が途中でちぐはぐになったりしてしまうね。残念だけど。

ところで、いつもおせわになっていたファンクションキー。今回は出番がないのかな、と思う君はためしてみないかな。「SELECT」機能の時は、「Ｆ４」は何も変化なし、「Ｆ１」と「Ｆ６」、「Ｆ７」、

●図15　BG画面指定エリア



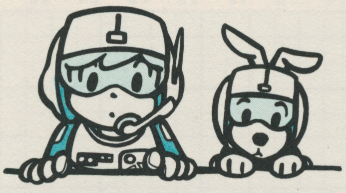
GAME BASIC

「F８」はRETURNキーと同じ。「F２」、「F３」、「F５」はスペースキーと同じ働きをするね。

「CHAR」機能の時にファンクションキーを押すと、あれ、いろんな文字が出てきた。これはあとで説明するから、お楽しみに。

さて、**「GAME BASIC」**に行く準備はいいかな。ESCキーを押して、**写真18**の状態でSTOP（ストップ）キー（**図12－⑥**）を押すと、**写真16**にもどる。ここで「GAME BASIC」を選ぶ、数字キーの「１」を押せばOKだ。

# 001-7



# キーボードのおさらい会

今までいろんな画面でいろんなキーをさわってきたけど、もう君は自由に使いこなせるかな。でも各ボードや BG 画面で同じキーが別の使われ方をしたり、少しややこしかったんじゃないかな。

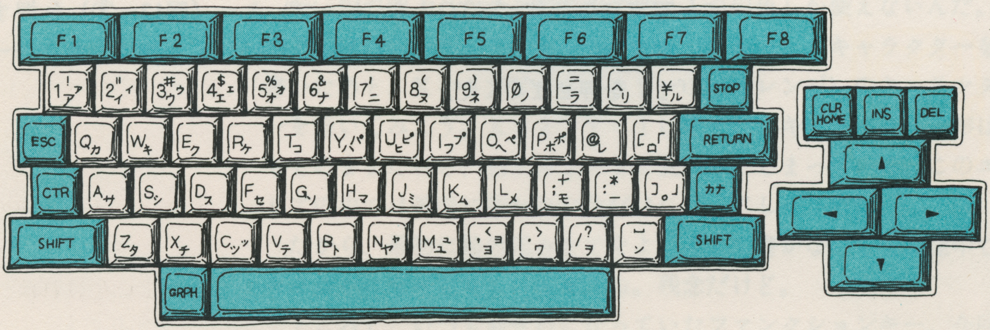
さて、いよいよ BASIC 画面にたどりついた。今までの画面と同じ使い方をするキーが多いけど、また違う使われ方をするキーもあるんだ。もう一度、おさらい会をかねて、BASIC でのキーの使い方を勉強しよう。

**図16**を見てほしい。

キーボードのキーを大きく分けると、次の2つがある。

**文字キー**
**特殊キー**

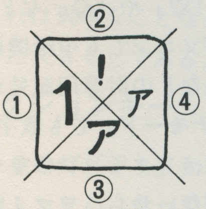
**●図16　キーボード**



# §1　文字キー

どれでもいいからキーを押してみよう。押したキーに書かれた文字が画面に出る。カナの使い方もメッセージボードの所で勉強したからわかるね。GAME BASICでも、この文字キーはまったく同じ使い方をするんだ。

●図17




●図18

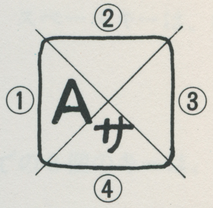
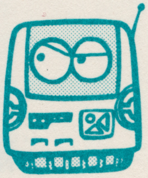


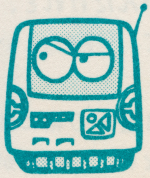

図17を見てほしい。一つのキーを対角線で区切ると4つの部分に分かれる。キーによっては4つの部分それぞれに、文字や記号のあるものと、図18のように、書かれていない部分のあるものがある。



英数字モード

「英数字モード」を覚えているかな。コンピューターに電源を入れたすぐあとはいつもこうなっている。この状態でキーを押すと、たとえば図17のキーなら「1」、図18のキーなら「A」が出るはずだ。つまり、**「英数字モード」**の時は、いつも対角線で区切ったエリアの①の部分が表示される。

「英数字モード」でSHIFTキーを押しながらキーを押してみよう。図17のキーでは「！」が表示されたけど、図18のキーでは何も表示されない。つまり、SHIFTキーを押しながらキーを押すと、②のエリアに書かれた文字が表示されているんだけれど、図18のキーでは、②のエリアに何も書かれていないから、表示されないというわけだ。



カナモード

カナキーを使って**「カナモード」**にすると、今度は普通の状態で③のエリアの文字、SHIFTキーを使うと④のエリアの文字が表示される。もちろん、そのエリアに何の文字も書かれていなければ、何も表示されないのは前と同じだ。

メッセージボードの所でやったように、「カナモード」の時に、SHIFTキーではなく、GRPHキーを押しながらキーを押すと、

「ガ」や「ダ」のような濁音が表示される。また、一番上の列の「1」から「ル」までの文字キーに限って、GRPH キーを押すことで、グラフィックキャラクターが出る。

この濁音と半濁音、つまり「バ」や「パ」の文字の表示のさせ方はファミコン特有なのだ。普通のパソコンでは、「″」と「°　」は別のキーに登録されていて、「バ」と一字ではなくて、「ハ」「″」と2文字になる。そのかわり、パソコンにあるGRPH キーと他のキーを押すと、「♡」や「♤」などのいろんなキャラクターが出てくる。

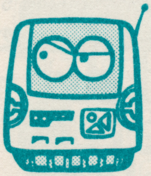
また、パソコンでも中級機以上になると、0から9までの数字や「＋」「−」などの計算用記号がキーボードの右側に独立してついていて、「カナモード」でも数字が打ち込めるようになっている。

その他は、ファミコンのキーの並び方はパソコンのものとほとんど変わりがないから、ファミコンでキーボードの使い方になれていると、いつかパソコンを買った時でも充分に使いこなせることになるよ。

## §2　特殊キー

文字キー以外のものを特殊キーというんだけれど、一つ一つ復習をかねて説明していこう。

### ①　SPACE（スペース）キー



**文字列**

本当は、このキーも文字キーの一つと言っても良いんだけれど、ただ、他の文字キーが何かの文字を表わすのと違って、「空白」という文字を入れるキーなんだ。

文字を入れる箱が3つある。はじめの箱に「A」、3番目の箱には「Z」が入っている。2番目の箱には何も入っていない。この箱が並んでいるものを**「文字列」**というんだ。じゃあ、この「文字列」の長さは、いくつ？　と聞かれたら何て答えるかな。2つ？　3つ？

たしかに、箱の中に入っている文字は「A」と「Z」の2個だけど、箱は3つあるね。だから、この場合は長さは3つ、というのが正解。それじゃ、文字列の長さが0というのはどんな時だろう。箱がないってことだ(むずかしく言うと、ヌル・ストリングスと言う)。

「ない」っていうのと、「空白がある」というのは違う。空白ということばが変なら、透明(とうめい)の文字と考えても良いだろう。

このスペースキーは、そういう透明な文字を表示する働きがあるんだ。プログラムを作る時は、スペースキーを良く使う。プログラムを見やすくしたり、コンピューターが感違(かんちが)いをしないように間をあけるためだ。普通の文章でも、「カネオクレ タノム」というのの「カネオクレタ ノム」と、間をあける場所が違うと意味が違ってしまうから、これは大事なことなんだ。それで、スペースキーは、押しやすい場所に大きくしてある。

## ② ファンクションキー

一番上部に並んだ「F１」から「F８」までの８個のキーは、これまでに何回もおせわになったね。

前にも言った通り、ファンクションキーには、それぞれ15文字以内（長さが15以内の文字列とも言えるね）の文字を登録できる。そして、そのファンクションキーを１回押すと、登録された文字列を全部押したのと同じ働きをしてくれるわけだ。

たとえば、スイッチオンのあと、「F３」を押すと、「コンニチハ」と表示されるのは、ファンクションキーの３番目に「コンニチハ」という文字列を登録してあったからだ。この時は、コンピューターがファンクションキーの中身を決めていてくれたのだけど、自分で中身を書きかえることもできるんだ。作戦指令(さくせんしれい)003-1で、もう一度くわしく説明しよう。

## ③ SHIFT（シフト）キー

このキーもおなじみだ。SHIFTキーを押しながら他のキーを押すと、違う文字を出すことができたね。

「SHIFT」とは、移すとか変えるとか交替(こうたい)という意味の英語だ。

## ④ カナキー

「英数字」モードと「カナ」モードの切りかえスイッチだ。中級以上のパソコンには、このキーの横に小さなランプがあって、今どっちのモードか一目でわかるものもあるよ。

### ⑤ GRPH（グラフィック＝GRAPHIC）キー

ファミコンでは、SHIFT キーと同じ使い方で、カナの濁音や、グラフィックキャラクターを出す時に使う。

### ⑥ STOP（ストップ）キー

BASIC では、プログラムを RUN（実行）すると、そのプログラムが終わるまでは、他のことはできない。実行中のプログラムを途中で止める時に使う。

また、何かを打ち込んでいて、このキーを押すと、カーソルは次の行に移って、今まで打ち込んでいた文字はコンピューターに伝わらなくできる。打ち間違って、直すのが面倒な時にも使えるよ。

### ⑦ ESC（エスケープ＝ESCAPE）キー

プログラムは普通、そんなに短いものじゃない。一画面で全部見えない時は、画面がかってに送られていってしまう。それを途中でちょっと止めたい時にこのキーを押す。STOP キーを押すと、そこで表示を完全にやめてしまうから、続きを見るには、また別の命令が必要になる。ESC キーで止めた時は、他のキーを押すと続きを見せてくれる。

### ⑧ RETURN（リターン）キー

テレビ画面に文字を出しただけでは、コンピューターは何もしてくれない。命令のことばを打ちこんだあと、この RETURN キーを押すことで、はじめて、ああこれは命令なんだな、とコンピューターがわかってくれる。また、プログラムを作っている時に、このキーを押すと、この行はこれで終わりだな、という意味になる。

コンピューターに命令を運ぶ役割のキーだ。

### ⑨ CTR（コントロール＝CONTROL）キー

このキーを押しながら、他のキーを押すことで、いろんな働きをする。くわしいことは「ステップ編」で説明しよう。このキーは、使わなくても良いのだけれど、あれば便利といったキーなのだ。

## ⑩ CLR・HOME（クリア・ホーム）キー

このキーには2つの働きがある。単独で押した時は、HOME（ホーム）機能、SHIFTキーを押しながら押すと、CLR（クリア）機能だ。

ホーム機能とは、画面の文字や記号を残したままで、カーソルだけを左上すみ（X＝0、Y＝0）の所に移す機能だ。この左上すみのことをホーム・ポジションということからこの名前がついた。

クリア機能は、画面に出ている文字や記号を全部きれいに消したあと、カーソルをホーム・ポジションへ持っていく。つまり、大そうじ機能だ。

## ⑪ カーソルキー

このキーも大活躍をしているね。上下左右にカーソルを移動する時にこのキーを使う。

キーボードでゲームをする時には、コントローラーの十字ボタンと同じように主人公を動かす時にもよく使う。ゲームのしすぎでこわさないように。

## ⑫ INS（インサート＝INSERT）キー

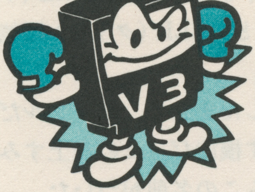
INSERTという英語には、間に入れるという意味がある。プログラムを入力していて間違った時に修正・編集（エディット）する時に使う。

## ⑬ DEL（デリート＝DELETE）キー

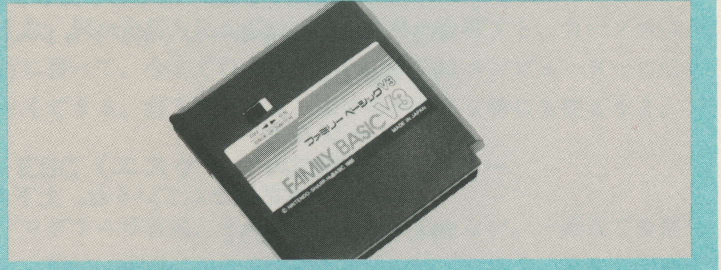
削除するとか、消すという意味があり、INSキーと共に、エディットに使う。

このINSキーとDELキーの使い方については、作戦指令003-2でくわしく説明しよう。

# 001-8




# GAME BASIC Ver. 3

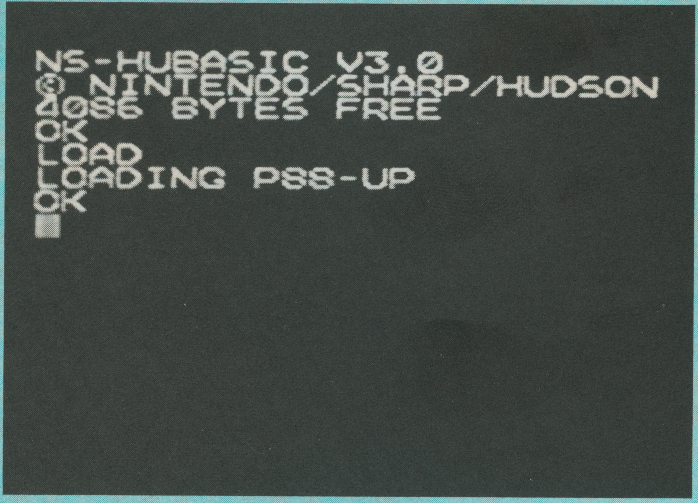



●写真21

ファミリー・ベーシックの新しいバージョンが発売された。まず、RAM（ラム）としてプログラムを記憶しておく場所が4Kバイトとなった。Ver. 2の2倍になったわけで、プログラムの大きさも倍のものが作れるわけだ。ということは，今までのVer. 2では作れなかった複雑なゲームを作ることができるわけだ。

命令語も、Ver. 2にくらべて26種類もふえた。プログラムを作る時に役立つ命令が多いけど、他にもBG画面についての改良がされているみたいだ。どんな命令がふえたのかは、このあと関係のある章でいっしょに説明していく。

まず、電源を入れてみよう。Ver. 2にあったオープニング画面や、占い、メッセージなどの画面は全部なくなった。BG画面も、Ver. 2ではBASIC画面と完全にわかれていたので、別々の画面が必要だったけど、Ver. 3では、BASIC上で直接操作できるようになったので、いきなり写真22の画面から始まる。Ver. 2では、「GAME BASICにしますか？　」に「ハイ」と答えたあと、「1～3」のうち、「1・GAME BASIC」を選んで、やっと出てくる場面だ。




●写真22

3行目を見てみよう。

**「4086　BYTES　FREE」**

となっている。Ver. 2では、

**「1982　BYTES　FREE」**

だったね。メモリーがふえたのがすぐわかる。

占いやミュージックボードがなくなったかわりに、ゲームが4つ入っている。写真22の状態で「F 1」から「F 4」のどれかのキーを押すと、それぞれのゲームがコントローラーで楽しめる。しかも、このゲーム、それぞれベーシックで書かれているから、ゲームをしている時にSTOPキーで実行を中止して、プログラムを書きかえることもできる。

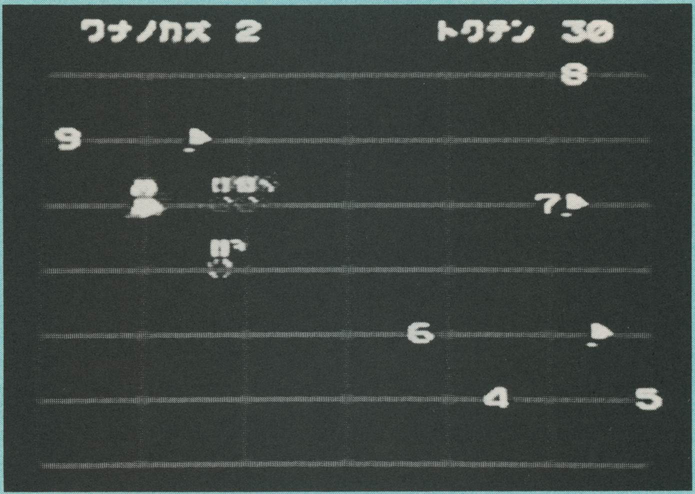
どこをどう変えたらどうなるかは、Ver. 3のカセットのケースに入っている「FAMILY BASIC V3 ハンドブック」にくわしくのっているよ。ベーシックをはじめたばかりの人には少しむずかしいかもしれないけど、この「ファミコン大作戦　ホップ、ステップ、ジャンプ」を読んだ人なら簡単にわかるはずだよ。

ところで、ファミコン本体のコントローラーIIに、マイクがあるのは知っていたと思うけど、どうやって使うのかは今までわからな

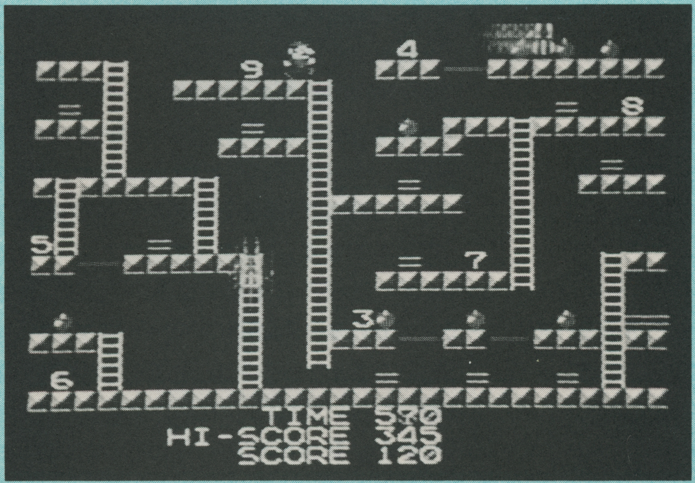
かったんじゃないかな。４つのゲームの内の１つ「ハート」は、このマイクを使ったゲームなんだ。マイクの使い方も、プログラムの中に説明されているよ。

ステップ・ジャンプ編では、このVer.３についても、ふかーく追求して行くのだ。

## バージョン３のゲーム



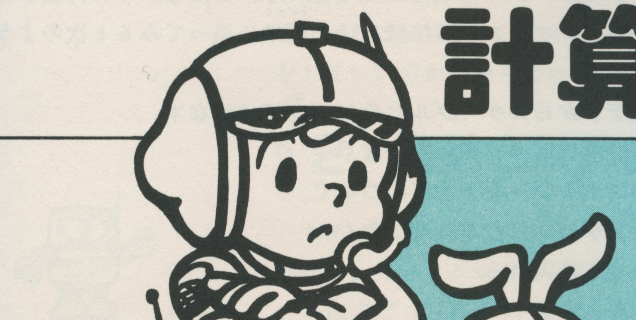

●写真23




●写真24

# 作戦指令 002

# BASICでの計算は？



# 002-1




# ダイレクトモードって何？

## §1　ダイレクト・モードとプログラム・モード

君はファミコンをテレビと接続する時に、どんな方法でやったかな。本体といっしょの箱に入っていた説明書を読みながらやったかな。それとも全部お父さんに接続してもらったのかな。もしかしたら、人に説明書を順番に読み上げてもらいながら、「この線は、ここにつなぐ」、「その線は、そこにつなぐ」と、一つずつ作業をしていったかもしれないね。

実はコンピュータに仕事をさせるのにも、こんなふうに方法がわかれるんだ。ファミコンの接続方法を簡単に追ってみると次のようになると思う。

**①アンテナからのケーブルとテレビを切りはなす。**

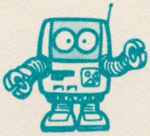
**②ケーブルとコネクターをつなぐ**

**③そのコネクターをテレビにつなぐ**

**④電源アダプターと本体をつなぐ**

**⑤カセットを入れる**

**⑥電源を入れる**



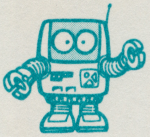
**手順**

少し順番に違いはあっても、だいたいこんなものだろう。このように一つのことをする時に、順番にやらなければならないことを「**手順**」という。

君は、この①から⑥までの手順を説明書を見ながら仕事をするのと、誰かに１つずつ読み上げてもらって、その言われた分の仕事をして行くのと、どっちが好きだろう。

コンピューターに命令をするのにも、この２つの方法があるんだ。

**プログラム**
**プログラム・モード**
**ダイレクト・モード**

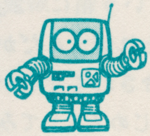


「手順」のことを「**プログラム**」と言い、その「手順」を書いた説明書を見ながら仕事をすることを「**プログラム・モード**での実行」と言う。一方、１つ１つ命令を受けながら仕事をすることを「**ダイレクト・モード**での実行」と言う。

このファミコンの接続をコンピューターにやらせたとしてみよう。まず、手順①を命令する。その時君が何か用事で、ちょっと出かけたとする。帰ってきてみると、コンピューターは①の仕事だけをすませたまま、じっと待っているだけだ。ところが、プログラム（手順を書いた説明書）をわたして、「これやっといて」と言って出かけて帰ってくると、接続は全部終わっているというぐあいだ。

コンピューターを使う時は、ほとんどこの「プログラム・モード」で行なうわけだけれど、何が何でも「プログラム・モード」なのかというと、そうでもない。たとえば、そのプログラムを作るためには、当然一つ一つ手で入力していくわけで、入力している時は「ダイレクト」で入力するしかない。でき上がったプログラムを保存したり、読み込んだりするのも「ダイレクト・モード」でしなければいけない。何よりも、プログラムができ上がったといっても、それを実行するには「ダイレクト・モード」で「RUN」という命令を出さなければならない。この命令が来るまでは、コンピューターは何をしていいのかわからないからなんだ。

# §2　変数と計算の記号



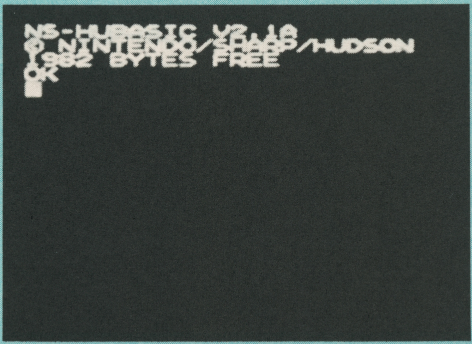
さあ、画面を見てみよう。今、GAME BASICにしたばかりの画面なら、カーソルが５行目の左（X＝０，Y＝４）の所で点滅しているはずだ（写真１）。その上には「OK」と出ているね。

**OK**

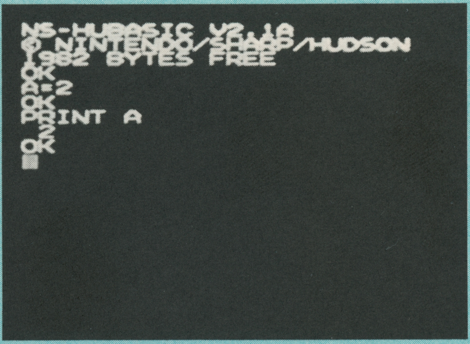
この「**OK**」は、「今、私は何もしていませんよ。いつでも命令をください。」というコンピューターからのメッセージなんだ。

**A＝2**

と入力してみよう。「２」のうしろで、カーソルが点滅している。今まで何回か説明したように、ただ入力しただけでは、テレビにその文字が出るだけだ。この命令をコンピューターに伝えるには、RETURNキーを押さなければならない。押してみよう。




●写真 1




●写真 2

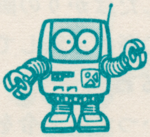
ただ「OK」と表示されて、その下の行にカーソルが点滅するだけだ。何も変わってないじゃないか。

では、次の行に、

**PRINT　A**

と入力して、RETURN キーを押してみよう。**写真 2** のようになったはずだ（「？SN ERROR」なんて表示が出たら入力ミスだ。もう一度やってみて！）。

最初の命令は、「A」という箱に「2」という数字を入れなさい、というもので、次の命令は、「A」という箱の中に入っている数字を画面に表示しなさい、という命令だ。最初の命令を与えた時に、なぜ何も表示しなかったんだろう。コンピューターは正確だけど、一つ一つ指示しないと、何もやってくれない。「「A」という箱に「2」という数字を入れなさい。というから入れたんだ。それがすんだから「OK」とサインを出したんだ。その他の事は、命令されてないからやらないよ。次にそれを見せろ、と言われたからはじめて見せてみせるんだ。」というわけだ（どこかのだれかに似てるみたい？）。



**変数**

この「A」という箱のことを「**変数**」と言う。もう学校で習ったかもしれないね。

＝

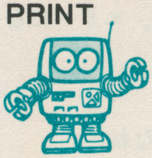
**A=A+1**

と入力してみよう。

ちょっと待った。「=」（イコール）の左（左辺）は「2」だし、右（右辺）は「2＋1」で3だから、式として間違ってるんじゃな

いか？。

**PRINT**



**PRINT A**

とやってみよう。ちゃんと「3」と出たでしょう。

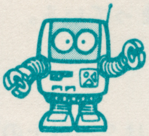
この「＝」、学校で習っているイコール、右と左が等しいですよ、という意味の他にもう一つの働きがあるんだ。右側（右辺）の数値を左側の変数に入れる（**代入**する）という働きだ。

**代入**

**A+1=A**

**SN ERROR**

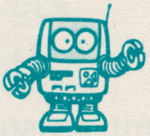
と入力してみよう。今度は「**?SN ERROR**」と出た。この「？SN ERROR」というのは、シンタックス・エラー（命令が間違ってますよ）という、コンピューターからの警告なんだ。なぜなら、「＝」の左側は変数じゃないからだ。つまり、「A＝A＋1」というのは、「A←A＋1」とでも書くと意味がよくわかると思うけど、この場合「＝」は計算のための記号というより、命令の一つなんだ。



**LET**

昔のコンピューターでは、これを「**LET** A＝A+1」と書いていた。けれども、いつの間にかこの「LET」という単語が省略されるようになったんだ。だから、「A＝A＋1」は、今「A」という箱に入っている数字に「1」を足したものを、もう一度「A」という箱に入れなさい、という意味だ。

次に、



**A=2.5**

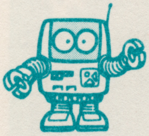
と入力しよう。「？SN ERROR」になったね。これは、ファミコンのベーシックが「**整数形ベーシック**」といって、ここで扱う数字は全部整数でなければいけない、と約束してあるからだ。変数には、整数しか入れる（代入）ことができないのに小数を代入しようとしたからなんだ。

**整数**

**A=5/2**

はどうかな。「PRINT A」とすると、「2」と出る。エラーにならなかった。カリキュレータボードのところで少しふれたように、コンピューターには「÷」と「×」という記号はなくて、「÷」は「／」、「×」は「＊」という記号を使っている。右辺は「5／2」で、「2.5」になるから、整数しか入らない箱に入れようとすると、前と同じでエラーになるはずじゃないか、と思う人もいるだろう。ところが、この命令では、先に「5／2」を計算する。その計算結果は、「整数

形ベーシック」だから、小数点の「.5」を無視して「2」になってしまうというわけ。「2」を「A」に代入するのだから、エラーにならないのだ。今度は、



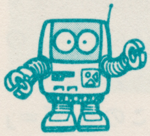
**A=(−5)/2**

として、「PRINT A」にしてみよう。「−2」になったね。

**A=65536**

**OV ERROR**

と入力すると、どうかな。「**?OV ERROR**」と表示された。これは「オーバー・フロー・エラー」といって、「扱える数の制限を越えています。」というエラーなのだ。



「整数形ベーシック」では、扱える数字の限度が決まっている。

**A=65535 : PRINT A**

としてみよう。

**:(コロン)**

ここで新しい記号「:」が出てきたね。この「:」は2つ以上の命令を一度に入力した時の区切り記号で「**コロン**」と言う。コロンで、一つ一つの命令を区切って、最後にRETURNキーを押すと、それぞれの場所でRETURNキーを押したのと同じ働きをする。プ

**ステートメント**

ログラムの中でも多く使われる。命令のことを「**ステートメント**」と言う。このコロンを使って、いくつもの命令を集めたものを「マルチ・ステートメント」と言うんだ。

さて、新しく覚えた「コロン」はいいとして、あれ?　「−1」と表示された。「65535」じゃない。

実は、これも「整数形ベーシック」の制約で、扱える数は「−32768」から「32767」までなんだ。「−1」から「−32768」までの整数は、32768個、「0」から「32768」までの整数も、32768個。合わせると、65536個。つまり、「整数ベーシック」では、65536通りの数しか扱えないんだ。ところが、0から65535までにすると、「マイナス」の数が扱えなくなる。そこで、

**65535=−1**
**65534=−2**
**∫**
**32768=−32768**
**32767=　32767**
**∫**

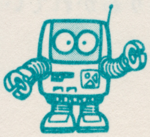
**1=1**

**Ø=Ø**

というように決めてあるんだ。人間にはわかりずらそうだけど、コンピューターには、この方がわかりやすいからだ。なぜかというと、コンピュータの中では2進法で計算されているからなんだ。(これについては、ジャンプ編でくわしく話そう。)とにかく、大きい数字を扱う時にはこのことに気をつけよう。

ところで気がついたかな。「－1」と表示された時は、その行の一番左（X＝0）が「－」と書かれているのに、「　2」の時は一番左が一つ空白（＝スペース）で、数字そのものは2番目（X＝1）の所に表示された。これは、次に出てくる文字変数との違いを出すために、数字を表示する時には、いつも「符号」(プラスかマイナス)をつけるためだ。ただ、正符号（プラス）の時は、それを省略して、かわりにスペースを使っているのだ。

# §3　文字変数

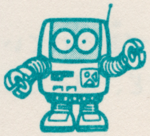


**CLS**

画面は今までいろいろ書いた文字や数字でいっぱいだ。

**CLS**

と入力してRETURNキーを押してみよう。画面がきれいにそうじされて、一番上に「OK」とだけ書かれた。この「CLS」という命令は、画面をきれいにする命令だ。キーボードのクリアキーとの違いは、クリア機能は命令ではないことだ。クリア機能では画面はきれいになるてカーソルが一番左上に出るだけ。「CLS」は命令だから、きれいにしたあと、命令終了のサイン「OK」を表示して、カーソルは2行目の一番左に出る。



**文字の代入**

変数の話しにもどろう。変数には数を入れる、ということはわかったけど、文字を入れることはできないだろうか。

**A$＝"TEST"：PRINT A$**

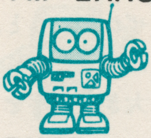
**文字変数**
**数値変数**
**"　"**

としてみよう。ちゃんと「TEST」と表示されるはずだ。この「A＄」は、**文字変数**といって、「＄」記号は「Aは**数値変数**ではなくて文字変数ですよ。」という意味を表わしている。

右辺に「"」「"」としてある。これは、「このマークで囲まれてい

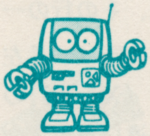
る部分は文字列です。」という意味だ。たとえば、「A＄＝AB」とやると、コンピューターは右辺の「AB」を数値変数としてしまって、「?TM EROR　」と返している。これは、タイプ・ミスマッチ・エラーと言って、右と左の形（タイプ）が合ってない、というエラーだ（数字を入れる箱を数値変数、文字を入れる箱を文字変数）。

**TM　ERROR**



コンピューターが、何かの数字を記憶する時、その記憶のために場所を使うわけだけど、その最小単位を**「バイト（BYTE）」**と言う。「1982　BYTES FREE」というのは、1982個分の場所（メモリー）があいているよ、という意味だったんだ。数値変数を記憶する時、整数型ベーシックでは、１つの変数につき、２バイトを使う。（逆に２バイトしか使わないから、扱える数値が決まって、小数も使えないという制約ができるとも言えるんだ）ところが、文字は１文字で１バイトを使う。そして、文字変数は、その中に31文字まで代入できるから、１つの文字変数は31バイト使ってしまう。そんなこともあって、文字変数と数値変数とは、はっきりわかれているんだ。

**BYTE**



**A$＝"1"：PRINT A$**

としてみよう。「1」となるね。

**A＝1：PRINT A**

ではどうだろう。「　1」となるはずだ。

数字も、文字の一部だということがわかるでしょう。前の式では、「1」は文字として扱われている。次の式では「1」は数字だから、数値変数の所で言ったように、プラスの意味の空白を１個作ったあとに「1」として表示されるわけだ。

この文字変数、一つの変数につき、長さが31文字まで許されている。また、文字変数と文字変数を足すことはできるけど、割ったり、かけたり、引いたりはできないよ。

**A$＝"TE"：B$＝"ST"：A$＝A$＋B$：PRINT A$**

で、「TEST」となる。

# §4　変数名はどんなもの

さて、この文字変数、数値変数は、いくつぐらい作れるだろう。

変数「ABCDEF」というのは可能だろうか。なんと、このGAME BASICでは、アルファベットではじまる、長さ255文字までの変数名が許されているのだ。実際には、そんな長い変数名は使わないし、メモリー自体、わずか1982バイトでは、プログラムに長い変数名を使うと、あっというまにメモリーオーバーだ。それに、この変数名、最初の2文字までしか区別しないんだ。

```
AAA=1 : AAZ=9 : PRINT AAA
```

とすると、「9」と表示されてしまう。「AAA」も「AAZ」も、最初の2文字は「AA」だから、コンピューターの中では同じ変数名とみなされて、あとで代入した数値「9」が入っているわけだ。

この変数名には、カナや記号は使えない。アルファベットと数字の組み合わせで、しかも先頭はアルファベットのものだけだ。また、ベーシックの命令語は使えない。

```
Aサ=2
A!=2
 IF =2
PRINT=2
```

などは、全部「?SN ERROR」となる。

ただし、

```
IFF=2
PRIN=2
```

などは許されている。また、「AA」と「A」は違う変数とみなされる。

```
1A=2
```

とすると、エラーこそ起きないけど、認められない。最初に数字があると、コンピューターが、これはプログラムの一部だと思ってしまうからなんだ。(63ページを見て!)

# §5 演算子

コンピューターでは、計算することを演算と言う。「演算子」というのは、計算する時に使う記号で、次の3つがある。

**算術演算子**

**関係演算子**
**論理演算子**

まず、算術演算子。これは、数を足したり引いたりする時に使う記号でおなじみのものだ。加減乗除、それそれぞれ「＋」「－」「＊」「／」を使う。「×」や「÷」はないことは前にも言った通りだ。分数を書く時も、「／」を使う。「$\frac{3}{8}$」→「3／8」というようになる。

除算のとき、「／」の右には0または、0になるような変数を使ってはいけないのは数学と同じだ。

もう一つ「MOD」というのがある。

**A＝5 : B＝2 : PRINT A MOD B**

結果は「1」となる。これは、「A」を「B」で割った残り（余り）という意味なんだ。

この「MOD」を使う時、左側が数字なら「5MODB」のようにスペースを入れなくてもいいけれど、文字の場合「AMODB」とすると、コンピューターが「AMDOB」という変数名だと思ってしまうので、必ずスペースをあけて「A MOD B」のように書かなければいけない。

関係演算子とは、2つの数や変数（文字変数もふくむ）を比べる時に使う。

| | |
|---|---|
| **＝** | **両辺が等しい　同じ** |
| **<>** | **両辺が等しくない（数学の「≒」）** |
| **>** | **左辺が右辺より大きい** |
| **<** | **左辺が右辺より小さい** |
| **>＝** | **左辺が右辺より大きいか、等しい（数学の「≧」）** |
| **<＝** | **左辺が右辺より小さいか、等しい（数学の「≦」）** |

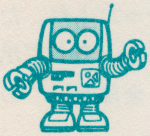
の6つがある。普通のパソコンは、

**< >→ >　,　>＝→＝>　,　<＝→＝<**

とも書くことのできるものがほとんどだけど、ファミコンでは、上の6つの方法しか許されていない。

ここでの「＝」は、今まで使ってきた「＝」とは使い方が違うのはわかると思う。

ゲームを作る時、君のうった弾が敵に当たったら、敵は死ぬ。と



**IF　THEN**

いうプログラムはどう作るだろう。敵のいる場所(本当はX＝，Y＝で表わすね）をAという変数に入れて、弾の位置をBという変数に入れると、

もし　AとBが同じ　ならば　敵は死ぬ

となる。これをベーシックで書くと、

**IF　A=B　THEN　敵は死ぬ**

となるんだ。

**IF　〜　THEN　□**

というのは、プログラムでよく使う命令語で、「もし〜の条件が成立したら□を行なうけど、成立しなければ、何もしない。」という意味だ。「〜の条件が成立したら」というのは「〜が真ならば」ということ。「〜の条件が成立しなければ」というのは「〜が偽ならば」とも言う。

ここで、コンピューターは「〜」が真か偽かをまず判断して、真ならば「**−1**」、偽ならば「**0**」という数字で返してくる。つまり、「IF〜 THEN」という文章は「〜」が−1なら「THEN」のあとの命令を実行して、0なら実行しない、という命令で、必ず「IF」と「THEN」は一組で使われる。

**A=(2=2) : PRINT A**

**A=(2>2) : PRINT B**

の二つをやってみよう。コンピューターはまず、カッコ（　）の中を優先する。ところが、カッコの中の「=」の両辺が定数だから、「=」は代入の意味ではなくて、両辺の関係を調べる記号だということがわかる。「2＝2」は、あたりまえで真なわけだから、「−1」を返す。次に「A＝」の「＝」は、左辺が変数だから、右辺のカッコの中、つまり「−1」をAに入れる。一方、下の式のカッコの中「(2＞3)」は間違っているので「0」を返し、Aに代入する。

**A$="YES" : A=2**

としてみよう。そのあとで、

**IF　A=2　THEN　PRINT　A$**

ちゃんと「YES」と表示されたでしょう。

**IF　A=3　THEN　PRINT　A$**

今度は何も出ない。そう、Aは3じゃないから、「THEN」のあとを

実行しなかったわけだ。

命令と命令の間には「：」を入れなければいけなかったよね。たしかにそうなんだけど、この「IF ～ THEN □」と、あとで出てくる「FOR ～ TO ～ STEP ～」文に限って、全部を一つの命令とみなすことができるのだ。

この関係演算子は、文字列・文字変数についても使える。

**"ABC"＝"ABC"**

もちろん、これならわかるね。それじゃ、大小はどうだろう。巻末のキャラクターコード表Bを見てほしい。コードナンバーとそれに対応する文字の表だ。「A」は65の所に書かれてある。「C」は67だ。もうわかったかな。「C」は「A」より大きいわけだ。この文字に対応するコードナンバーが大きいほど、その文字が大きい。２文字以上の文字列はというと、まず両方の一番最初の文字を比べる。違う文字なら、その１文字のコードナンバーの大きい方が大きい文字列。もし同じ文字なら、それぞれの二番目の文字を比べる。そうやってずっといっしょなら、長い方が大きいというわけなんだ。

**"C"＞"A"**

**"CCC"＞"CCA"**

**"CAA"＞"CA"**

と、いろいろ実験してみよう。

最後に、論理演算子。これには４つある。

**AND**（論理積）
**OR**（論理和）
**NOT**（否定）
**XOR**（排他的論理和）

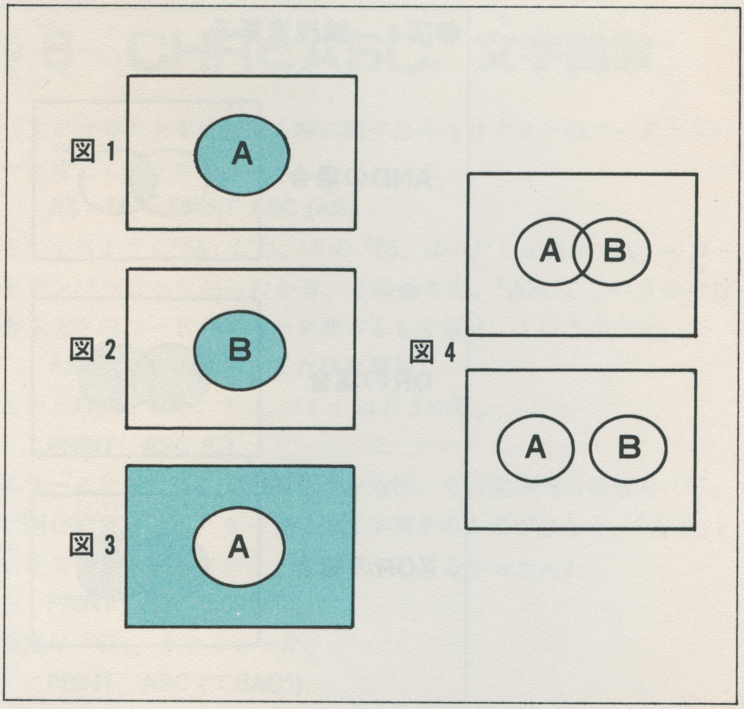


図1
図2
図3
図4

100人の人がいて、数学と国語の試験を受けたとしよう。この内、数学で50点以上をとった人(数学の点数＞＝50の条件を満たす人だ)をA集団とする。一方、国語の50点以上の人をB集団とする。それぞれ、図1と図2。四角いワクは100人を表わしたもので、中の斜線の入った円は、A集団、B集団だ。

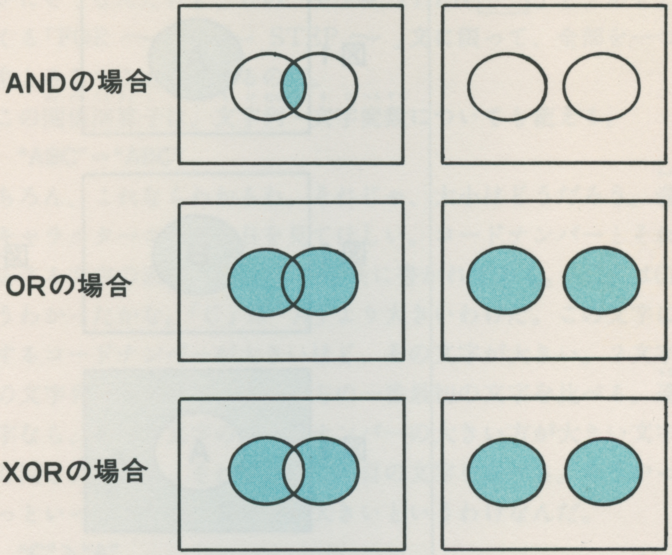
この時、「NOT　A」の集団と書くと、図3になる。つまり、条件を満たした集団Aでない他の人たち、または、Aという条件に合っていない集団ということだ。次に、このAとBを重ね合わせてみよう。両方の科目で50点以上をとった人がいれば上の図に、いなければ下の図になる(図4)。

「A　AND　B」は、Aでもあるし、Bでもある人たちの集団。

「A　OR　B」は、Aか、またはBの集団。

「A　XOR　B」は、Aか、またはBの集団の中で、A、B両方の条件の人を除いた集団、つまり、Aだけか、またはBだけに合てはまる集団（図5)。

●図５　論理演算子




それぞれの斜線の部分が、「AND」「OR」「XOR」の条件を満たす集団だ。

ゲームのキャラクターは、Ｘ座標と、Ｙ座標の２つの変数で、どこにいるかを表わすのが普通だ。とすると、弾が当たったかを調べるのに、次のように使う。

**IF　AX＝BX　AND　AY＝BY　THEN　死ぬ**

**AX→　キャラクターのＸ座標**<br>
**AY→　キャラクターのＹ座標**<br>
**BX→　　　弾　　　のＸ座標**<br>
**BY→　　　弾　　　のＹ座標**

この論理演算子は、他にも使い方がある。ジャンプ編でもう一度追求してみよう。

演算子とか論理積とか、難かしい日本語がいっぱい出てきたけれど、こんな言葉覚えなくってもいいんだ。ただ、それぞれのだいたいの働きだけは頭に入れておいてほしい。

## §6 CHRとASC, 文字関数

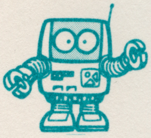
文字列の大小を比較する時に調べたキャラクターのコードナンバーを覚えているかな。

ASC(　)

**A$＝“A”：PRINT ASC (A$)**

としてみよう。「65」だね。この「65」は、たしか文字「A」の**コードナンバー**じゃなかったかな。その通りだ。「ASC」という命令は、ある文字のコードナンバーを数字として直せ、という命令だ。

コードナンバー



**ASC（目的の文字、または文字列）**

という形式で書く。では、「C」はどうかな。

**PRINT　ASC (C)**

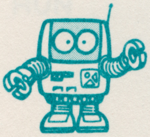
エラーになったね。文字列を扱う時は、その文字列の前後を「“”」で囲む約束だった。さっきは、文字列そのものではなく、「A＄」という文字変数だったから「“”」は、いらなかったんだ。

**PRINT　ASC (“C”)**

今度は「67」。うまくいった。

**PRINT　ASC (“CBAD”)**

やはり「67」だ。カッコの中の文字列の長さが2以上の時は、一番最初の文字のコードナンバーを与えてくれる。



今度は逆に、コードナンバーで文字を書くことができるかどうかやってみよう。

CHR$(　)

**PRINT　CHR$ (65)**

これで、「A」という文字が出たね。カッコの中に0から255までの数字を入れると、そのコードナンバーの文字が出てくる。カッコの中に、マイナスや、256以上の数字を入れると、そんなコードナンバーはないから、エラーになってしまう。もちろん、

**A$＝CHR$(67)：PRINT　A$**

のような使い方もできる。

## §7 プログラムの書き方

今まですっと、「ダイレクト・モード」での入力を続けてきた。

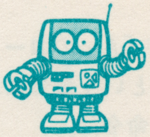
```
D=Ø : PRINT  D
```

答えは、もちろん「0」だ。では、

```
1Ø  D=8Ø : PRINT  D
```

としてみよう。今度は何も表示しないし、「OK」のサインも出てこない。たしか、

```
1A=2
```



LIST

とやった時も同じだったはずだ。ついでにこれも入力してみよう。どうしたんだろう。

実は、これがプログラムなんだ。ためしに、

```
LIST
```

と入力してごらん。

```
1  A=2
1Ø  D=8 : PRINT  D
```

となったはずだ。

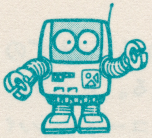
48ページのファミコン接続の手順を思い出そう。①とか②とか、頭に数字があって、その順番通りにするんだったよね。ここの「1」とか「10」というのは、この①，②と同じ意味なんだ。「1」とか「10」というのを行番号という。なぜ「1」の次は「10」なのかって？ これは別に「2」でも「10000」でもかまわないんだ。

```
5  RPINT  A
```

としてほしい。そのあと、

```
LIST
```

とすると、



```
1  A=2
5  PRINT  A
1Ø  D=8 : PRINT  D
```

行番号

となっている(この「1」とか「5」を**行番号**という)。

ファミコン接続の手順の時、③と④の間に何か別の命令をはさもうとすると、その命令が④になって、もとの④は⑤に、⑤は⑥にと全部ずらしていかなければならないね。

ところが、こうやって行番号の間をあけておくと、今やったように、あとからその間の行番号をつけた命令を入れると、コンピューターが自動的に小さい順に並べかえてくれる。そして、行番号をず

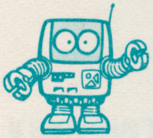
らしていく必要もないね。行番号は、0から65535まで許される。(数の制限と同じだ)

では、なぜ「OK」が出ないんだろう。「OK」は命令を与えて、実行が終わった時にコンピューターが返してくるものだった。ところが、今君がやっているのは、プログラムを作っていることで、実行は命令していない。だから、実行が終わったサイン「OK」も返ってこないんだ。

行の一番最初が数字だった時に、RETURN キーを押すと、それは、プログラムを記憶する場所へ送りこむしるしになるんだ。そして「LIST」という命令は、プログラムを行番号の小さい順に並べて見せてくれ、という直接命令で、「LIST」のあとにすぐ押したRETURN キーは、その行のはじめが数字でないから、プログラムじゃなくて、命令だと判断する。そして、記憶する場所へ送るのではなくて、コンピューターに伝えるためのしるしになるんだ。

**1A=2　→　1　A=2**

となったのも、行番号と命令を分けるために、コンピューターが、1字分を空白にして見やすくしているのだ。



**RUN**

このプログラムを実行させる命令が、

**RUN**

だ。「2」、「8」と表示されたあと、ちゃんと「OK」が出るでしょう。プログラムの中の命令を順番に実行したから、「2」と「8」を表示して、実行終わりの合図に「OK」を出してきたのだ。

**PRINT　A**

とダイレクトモードで聞いてみよう。ちゃんと「2」になってるね。

**5　A=5：PRINT　A**

として、

**LIST**

と命令しててみよう。(これを、「リストをとる」という)

**1　A=2**

**5　A=5：PRINT　A**

**1Ø　D=8：PRINT　D**

になっている。

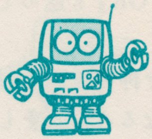
同じ行番号でプログラムを入力すると、あとで入力された行がプ

ログラムの中に入るしくみだ。

さて、ここでもう一度

```
PRINT A
```

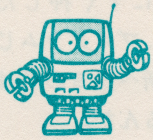
にしてみる。あれ、今度は「0」になってしまった。さっきは「2」だったし、今入力したのはプログラムの一部で、それに「A＝5」だったはずなのに。



これは、プログラムの一部を変えたり、新しく入力すると、すべての変数の値は0になる（登録をとり消される）という約束だからだ。(一部のパソコンを除く）このことを、変数を**「クリアする」**と言う。ここで一度「RUN」したあと、

**数値のクリア**

```
PRINT A
```



とすると、今度は「5」になっている。「RUN」したあと、プログラムを変更していないから、変数「A」の値が保存されているわけなんだ。ここで、

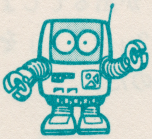
**CLEAR**

```
CLEAR
```

とダイレクトで命令したあと、

```
PRINT A
```

としてみよう。あれ、プログラムを変更してないのに、Aの値は「0」になっている。この「CLEAR」という命令は、すべての変数を0にする、つまり、「クリア」にしろ、という命令なんだ。



言いかえれば、プログラムの一部を変更したり、「RUN」を命令すると、自動的に(君が命令しなくても)、このCLEARの命令を実行する、とも言えるのだ。

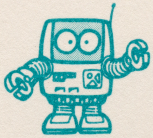
**NEW**

```
NEW
```

とダイレクトで命令してみよう。「NEW」。新しい、という意味だけど、どうなっただろう。

```
LIST
```

と命令して見てみよう。と思ったけれど、リストが出ない。なんとプログラムが消えてしまったのだ。



この「NEW」という命令は、プログラムを全部消して、それに変数も全部消してしまう、という命令だ。こうなると、普通の方法ではもう、もとのプログラムを呼び出すことはできない。ここで、

**FRE**

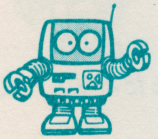
```
PRINT FRE
```

としてみよう。「1982」となるはずだ。この「1982」という数字は、どこかで見たね。そう、GAME BASICにした一番最初の画面の3行目に、

**1982 BYTES FREE**

と出てたね。

プログラムも変数も、全部消してしまったので、一番最初と同じぐらい、記憶する場所ができたということだ。そして、この「FRE」というのは、残っている記憶できる場所の数を意味する変数なんだ。



普通、変数は「A＝2」というように、君がその値を与えなければいけないんだけれど、この「FRE」のようなものは、コンピューターが自動的にその値を決めてくれる、「**予約関数**」とでも言えるものだ。

**予約関数**

# §8 ファンクションキー

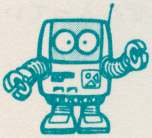
というわけで、今、何もかもクリアされて、一番最初の画面にもどったところで、

**KEY LIST**

**KEY LIST**

と命令してみよう。写真3のようになったはずだ。

ファンクションキーについては、君はもう充分知っているね。「LIST」というのはプログラムを見せなさい、という命令だったはずなのに、この「KEY LIST」というのは何だろう。



「KEY LIST」とは、ファンクションキーのそれぞれにどんな文字列が登録されているか見せなさい、という命令なのだ。

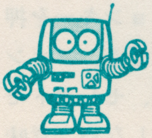
ファンクションキーの8つのうち、「F 2」、「F 4」、「F 7」、「F 8」はもう知っているね。「F 5」の「**SPRITE**」は作戦指令003で、「F 1」の「LOAD」はこのあと002-4で、くわしく説明する。

**SPRITE**

「F 3」の「**GOTO**」はなんだろう。これはプログラムの中で、

**GOTO**

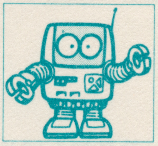
**GOTO 3ØØ**



のように使う。プログラムは、行番号の小さい順に実行されるんだけれど、時々飛び越して他の所を実行させたり、今実行されている行の番号より小さい行番号の所に書かれた命令を実行させたい時がある。そんな時に使う命令で、普通はプログラムの中でしか使われ

ない。くわしい実例は作戦命令003でたくさん出てくるはずだ。

## CONT



「Ｆ６」の「**CONT**」というのは、逆に、ダイレクトモードでしか使わない命令だ。プログラムを実行中にSTOPキーを押すと、プログラムの実行が中止される。というのは、キーボードの所で説明したね。中断した所からそのプログラムを再開したい時に、この命令を使うんだ。

さて、「Ｆ４」を押してみよう。

**CHR$ (**

と出て、「(」のすぐうしろでカーソルが点滅しているだろう。一度STOPキーを押したあと「Ｆ２」を押すと、

**PRINT**

と出て、「T」から一字おいてカーソルが点滅している。またSTOPキーを押して、今度は「Ｆ７」を押してみよう。

**LIST**

と表示されて、次の行に「OK」が出た。そのもう一つ下の行にカーソルが出ている。

PRINTの時はすぐわかるね。「Ｆ２」に登録されているのは、「PRINT」という文字列ではなくて、「PRINT　」と、「T」のうしろにスペース１文字分が入った長さ６の文字列だったんだ。それに比べて、「Ｆ４」の「CHR$ (」は、見た通り長さ５の文字列が入っていたわけだ。

「Ｆ７」は何だろう。この状態は、プログラムがぜんぜん入っていない時に、

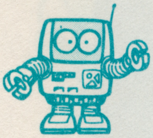
**LIST**

のあとRETURNキーを押したのと同じ状態だ。

実は、このRETURNキーも、画面には表示されないけど、一つの文字として立派にコードナンバーを持った文字ともいえるんだ。つまり、「Ｆ7」には、「LIST」にRETURNキーという合計5文分の文字列が登録されていたのだ。(「(M)」と最後に書かれているもの)

ここで、BG画面の「セレクトモード」でファンクションキーを押した時のことを思い出してほしい。実は、あのモードでも、ファンクションキーの中身は今と同じだったんだ。ところが、セレクトモードでは普通の文字キーは、いくら押しても何も変化しない。意味

のあるのは、スペースキーとRETURNキーだったね。だから、スペースキーをふくむ文字列が登録された「F 2」「F 3」「F 5」は、スペースキーと同じ働きをし、RETURNキーをふくむ文字列が登録された「F 1」「F 6」「F 7」「F 8」は、RETURNキーと同じ働きをした。そして、画面のキーをふくまない「F 4」は、何も変化を起こさなかったんだ。



さて、ここで何回も話した、ファンクションキーへの文字列の登録法を説明しよう。

```
KEY 1,"LIST"
```

と、ダイレクトで打ち込んでみよう。そのあと、「KEY LIST」と入力すると、お見事、「F 1」は「LIST」に変わっている。「F 1」キーを押すと、

```
LIST
```

と表示されて、カーソルは「T」のとなりで点滅している。「“”」で囲まれた文字列の「T」のあとに、スペースを定義していないからあたりまえだ。スペースを入れたい時には、

```
KEY 1,"LIST  "
```

とすれば良い。

この「KEY」のあとの「1」を「2」から「8」へ変えることで、「F 2」から「F 8」の内容を登録できるわけだ。

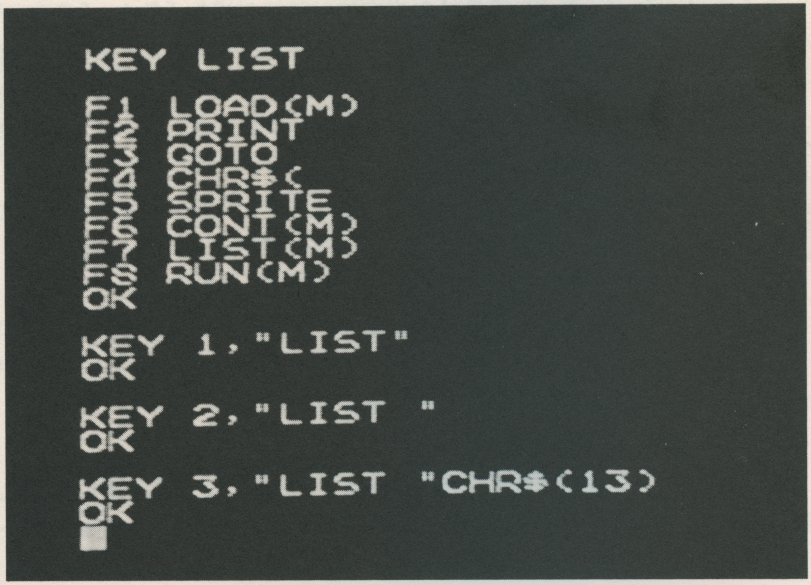
それでは、RETURNキーを登録するにはどうすればよいのだろうか。RETURNキーのコードナンバーは巻末のキャラクターコード表にはのってないけど実は13なんだ。そして文字列の足し算ができる、という説明を思い出そう。

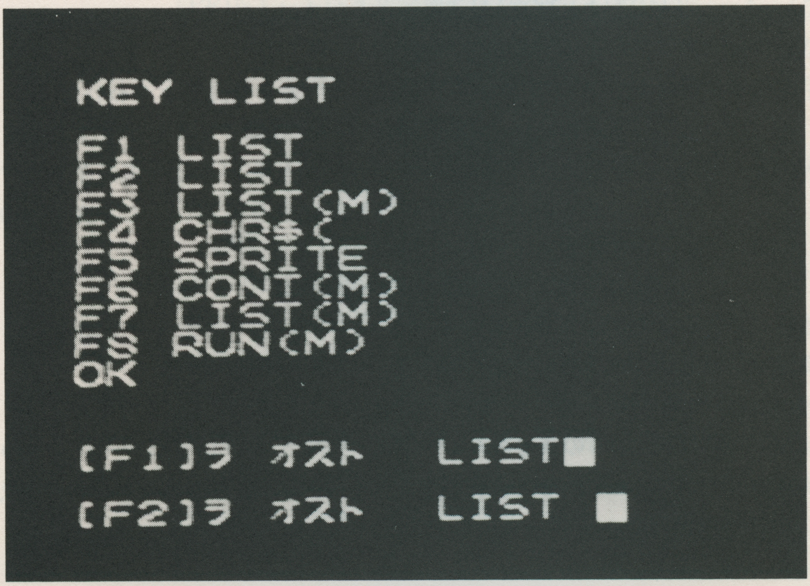
```
KEY 1,"LIST"+CHR$(13)
```

これで「F 1」キーは、「F 7」キーとまったく同じになったはずだ。テストしてみよう。

この章を読んでいる君、これでもういくつベーシックの単語を覚えたかな。数えると、もう23個にもなるんだ。もう一息だよ。

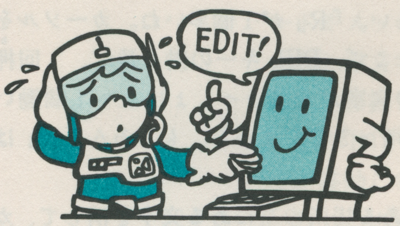
ファンクションキーへの登録



# 002-2

# まちがえたでも大丈夫

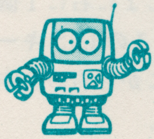



さて、今までいろいろ打ち込んできて、君は何回ぐらい「ビーッ」という音と「？SN ERROR」という表示を見てきたかな。

この「？SN ERROR」は、前にも言った通り、命令が違う、文法が違うという意味だ。

```
1Ø  A$=CGR$ (65 : PRINRT A$
```

と、まずプログラムを入力してみよう。見ればすぐわかる通り、この１行は、間違いだらけだ。でも、とにかくRETURNキーを押す。この段階では、まだエラーは起きないね。ここでのRETURNキーは、プログラムを記憶する場所に、この行を送り込むという役割しかなく、この命令を実行させたわけではないから、コンピューターにもエラーがあるかどうかわからないんだ。



IN

さて、「RUN」させてみよう。ダイレクトで「RUN」と打って、RETURNキーを押すか、「Ｆ８」を押すだけでも同じだったね。

```
?SN  ERROR  IN  1Ø
```

と表示されただろう。この「IN 10」は、プログラム実行中にエラーが出た時、どの行にエラーが出たかを教えてくれるメッセージで、この場合10行にエラーがあるという意味だ。

```
LIST
```

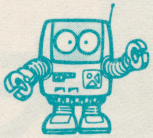
をとってみよう。まず、「CHR$」の「H」が「G」になっている。カーソルキーでカーソルを「G」の文字に乗せよう。その位置で「H」のキーを押す。ちゃんと「CHR$」になって、カーソルは「R」の所にある。ここでRETURNキーだ。今、カーソルは「OK」の所にあるね。カーソルを一つ下にして、またリストをとってみよう。

```
1Ø  A$=CHR$ (65 : PRINRT  A$
```

となってるね。RETURNキーは、いつもその行の一番最後の所で押

さなくても、続いている行なら、そのどこででも自由に押すことで役目をはたしてくれる。

## DELキー

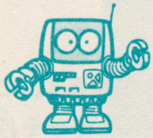


次に、「PRINRT」がおかしい。「R」が1個多いね。カーソルを「T」の所に持っていこう。ここで、DELキーの働く番だ。1回押してみよう。「T」からあとの文字が全部いっしょに左へ1個働いて、「T」は「R」のあった場所へきてしまった。もちろん「R」は消えてしまった。

DELキーは、今カーソルがある場所の左にある文字を消して、カーソルからうしろにある文字を全部引きつれて、動かしてくれる役割を持っているんだ。

リストをとってみよう。

```
1Ø  A$=CHR$ (65 : PRINT  A$
```



もう1個間違っているね。「CHR$」はカッコ「(　)」の中に数字を入れて、「CHR$ (65)」のように書くのが約束だ。ところが、このプログラムでは、とじるカッコ「)」がない。

## INSキー

またカーソルを、今度は「：」の所に持っていく。ここで、INSキーを1回押してみよう。どうなったかな。DELキーとは逆に、カーソルの所の文字から右側にあった文字列が全部いっしょに右に1個ずれて、「5」と「：」の間にスペースが1個できた。このINSキーは、カーソルのある場所に1個すきまをあけて、そこから右の文字列をその分移動する。DELキーとは逆の役割なんだ。

さて、カーソルは、今あいた空白の所にあるから、そのまま「)」を打ち込む。これでOKだ。RETURNキーを押して「RUN」させよう。ちゃんと「A」が出るね。

実験だ。今「OK」の1つ下の行の一番左でカーソルが点滅しているね。INSとDELキーを押してみよう。何か変化があったかな。何もないね。この行には、まだ何も書かれていないから、INSもDELも、その役割を果たそうにも、果たせないのだ。DELキーが働くためには、カーソルの左に文字がなければならないし、INSキーが働くためには、カーソルの所か右に文字がなければならないんだ。

さて、もう一度実験。リストをとって、カーソルを「：」の所に合わせよう。そして、INSキーを押してみよう。今度は何回もだ。一回押すごとに、空白がどんどんふえていく。カーソルは、最初の

「)」のうしろにあるね。「A＄」の「＄」が、この行の一番はしまできてしまった。それでも１回押してみよう。前のミュージックボードやメッセージボード、カリキュレータボードでは、こうすると「＄」の文字が消たんだよね。ところが今度は「＄」は次の行の一番左へ移動して、下の「OK」の文字が一段下へずれてしまった。どんどんINSキーを押してみよう。いくらでも「：PRINT A$」の文字が移動して、下へ下へと降りて行く。「OK」の文字は一番下の行へ押しこめられて、とうとう画面から消えてしまった。「：PRINT A$」は画面の右下すみまで来た。まだ押して見よう。「＄」は右下すみ（X＝27，Y＝23）から、左下すみ（X＝０，Y＝23）へ移り、「：PRINT A」は下から２行目の右すみへ上がった。画面に残っている他の文字も全部１行上に上がった。

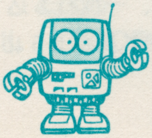
今度は、そのままカーソルを「：」の所に持ってきて、DELキーを押してみよう。メッセージボードなどでは、カーソルがその行の一番左まできたら、カーソルのうしろの文字まで消えていったね。でも今度は、上の行へ上の行へとうしろの文字を連れたまま、行を乗り越えて行く。それでもDELキーを押し続けていく。とうとう「CHR$ (65)」の文字も消してしまう。

**1Ø：PRINT　A$**

の所で止めて、一休み。カーソルは「：」の所だね。DELキーを、あと４回押してみよう。

**RINT　A$**

「Ø」を消して「1」を消して、もう左側に消すものがなくなって、次に、カーソルの位置にあった「：」を消して「P」を消した。今カーソルは「R」の所にある。やっとメッセージボードの時のようになり出した。



**ベーシックの１行**

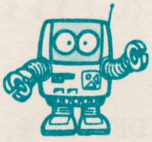
これは、メッセージボードなどでは、エディットの時の「１行」とは、画面上の「１行」だったけど、**ベーシック**での「**１行**」とは、画面上の「１行」ではないということだ。

少ないメモリーでプログラムを作るには、どうしても「：」を使ったマルチステートメント（覚えているかな）が多くなる。ところが、ファミコンは１行が28字しかない。そこで、ベーシックでは、画面上の行を越えても１つの命令のつながりを（つまり、切れ目の

サインであるRETURNキーを押すまでは)、全部1行として扱われる。

だから、INSやDELキーの働きも、ミュージックボードの時と違って、ベーシックでの1行、つまり画面上では何行にもわたっている1行に対して行なわれるのだ。画面上に何行にもわたっているプログラムなのに、行番号は1個しかない、というのもこれでわかったかな。でも、ベーシックの1行が何行にもわたって書けるといっても、無制限につくら長くてよいわけじゃない。長さは255までという制限がある。(この255というのは、命令の数でもなく、文字の数でもなくて、コンピューター内部の表現のしかたで数えているので、ちょっとむずかしい。ジャンプ編で説明しよう。)

長い長いプログラムを1行としてRETURNキーを打ち込むと、あとでリストをとった時、うしろの方に打ち込んだはずの命令が欠けていたりするのは、この制限を越えたせいだ。でも、ミュージックボードなども、ベーシックと同じように、全部の行がつながっていてくれた方がやりやすかったと思いません?



**スクロール**

ところで、INSキーをずっと押していった時に、画面が全体に1行上がって、一番上の行は消えてしまったのに気がついたかな。画面の一番下まで書いて、次に何かを書こうとすると、その分の場所をとるために、全体が1行上がってしまうわけで、このことを「**スクロール**」というんだ。ゲームによっては、このスクロールを利用することもあるし、逆にスクロールが起こるとこまることもあるので、この現象は覚えておこう。

とにかく、このINSとDELキーとカーソルキーで、一度打ち込んだものを自由に編集、修正(エディット=EDIT) ができるようになったでしょう。

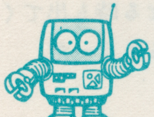
# 002-3



# エラー！エラー！エラー！

「？SN ERROR」というのはもう何回もお目にかかったね。また、プログラムの中でエラーが起きた時は、

**SN ERROR**

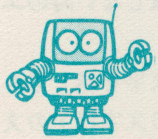


**?SN ERROR IN 1Ø**

のように、どこで間違いがあったかを行番号で教えてくれるのも知ってるね。

エラーを起こすことは、ぜんぜんはずかしいことじゃない。何10年もプログラムを組んでいるベテランでも、新しく作ったプログラムが一発で正しく走ることはまずないだろう。エラーを出して、直して、またエラーを出す。プログラム作りは、エラーとの闘いだ。スキーやスケートが何回もころんでるうちに上手になるように、プログラム作りもエラーをこわがってちゃうまくならない。

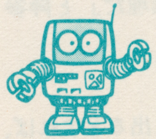
一口にエラーと言ってもたくさんあるんだ。ファミコンでは、図の通り、17種類ある。「SN」エラーはもう説明もいらないだろう。今までに勉強してきたベーシックで出てくる可能性があるのは、「SN」を入れて９種類だ。



**PRINT CHR$ (342)**

**IL ERROR**

これを入力して出てくるのが「IL」エラー（イリガル・ファンクション・エラー）だ。「CHR$」は、コードナンバーから、その文字を呼び出す命令だった。ところが、コードナンバーは０から255までしかないのに342番を呼び出そうとした。つまり、決められた呼び方（０から255まで）の約束を破った、というエラーだ。

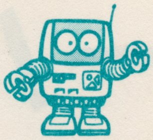


**PRINT 3ØØ*3ØØ**

300×300は90000だね。ところが、「整数型ベーシック」では、数の制限があったはずだね。90000は、その制限を越えてしまっている。「OV」エラー（オーバーフロー・エラー）という。ここで気を

**OV ERROR**

つけなければいけないのは、53ページでも説明したように、エラーが出なくても、計算結果がおかしくなることがある。

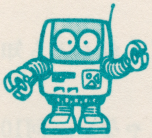


PRINT 2ØØ*2ØØ

200×200は、40000ではなく−25536になる。なぜそうなるかはページの説明の通りだ。気をつけよう。

**OM ERROR**

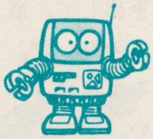
次のエラーは「OM」エラー（アウト・オブ・メモリー・エラー）だ。最初に説明したように、ファミコンは、Ver. 2で1982バイト、Ver. 3でも4086バイトしかメモリーがない。それに、あいているメモリーは、プログラムを入れる所と、変数を覚えさせる所との両方に使っている。だから、プログラムを作っていて、プログラム自体はメモリーにおさまっていても、いざ動かそうとすると、プログラムの中に使われている変数の値を入れる場所がなくなる時も出てくる。これからプログラムを作る時、一番頭が痛くなるエラーだ。



PRINT 5/Ø

**DZ ERROR**

これは単純なエラー、「DZ」（ディヴィジョン・バイ・ゼロ）つまり、「Ø」による割り算だ。何かの数を0で割ると、無限大になってしまう。割りきれないわけだ。変数を使っていて、その値が0になる可能性があるものは気をつけよう。



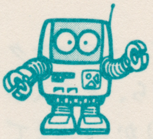
A$=3

**TM ERROR**

これも何回か見たかもしれない。「TM」（タイプ・ミスマッチ・エラー）。右辺は数値、左辺は文字変数、両辺の型が一致しない時のエラーだね。

**ST ERROR**

「ST」（ストリング・トゥ・ロング）。文字入力が31文字を越してしまった、というエラーだ。ただし、文字変数に直接31文字以上の文字列を入れようとすると（A＄＝“31文字以上の文字”）、「IL」エラー。文字列の足し算の結果が31文字以上になるとき（20文字の文字列＋20文字の文字列）は、「OV」エラーにしかならない。



**FT ERROR**

「FT」（フォーミュラ・トゥ・コンプレックス・エラー）。カッコなどが多すぎて、コンピューターが「私の能力じゃ無理だ」と降参する時のエラー。たとえば、

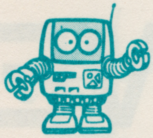
A=3+4 * (5−2)

こんな式で「FT」エラーが出るわけはないけれど、もし出るとすれば、まず次のリストを入力してみよう。

A=5−2

A=3+4 * A

のように、もとの式を分けてやることで解決できる。



**CC ERROR**

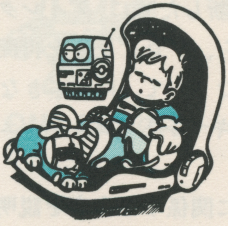
プログラム実行中にSTOPキーで実行を中断した時に、ダイレクトで「**CONT**」と入力すると、中断した所から、プログラムを再開できるんだ。でも、プログラムの一部を変えたりして、再開できなくなった時、「CONT」を命令すると、「**CC**」(キャント・コンティニュー)と言って、エラーになる。

さて、残りのエラー9種類は、そのエラーに関係する命令を説明するところでくわしく話そう。

エラーをこわがらないで。このエラー・メッセージはコンピューターが教えてくれるヒントなんだから。

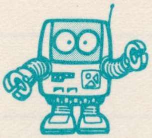
| 表示 | 意　味 |
|---|---|
| SN | 文法や単語のつづりがまちがっている。 |
| IL | 規定以外の数値を使って命令や関数を呼んでいる。 |
| OV | 計算結果が数値の制限をこえた。 |
| OM | メモリーが足りなくなった。 |
| DZ | θで割り算をした。 |
| TM | 右辺と左辺で変数の形(文字と数値)が違う。 |
| ST | 入力する文字が31文字をこえた。 |
| FT | (　)が多すぎる。式が複雑すぎてお手上げだ。 |
| NF | FORがないのにNEXTがある。 |
| TP | カセットテープからの読み込みに失敗した。 |
| UL | GOTO、GOSUBの行き先き行番号がない。 |
| RG | GOSUBがないのにRETURNがある。 |
| OD | READで読むDATAの数が足りない。 |
| MO | パラメーターの必要な命令に、その指定がない。 |
| CC | プログラムの再開(CONT)ができない。 |
| SO | 指定してない配列変数や、指定以外の添字を使っている。 |
| DD | 同じ配列変数を2重に指定している。 |

# 002-4



# テープへSAVE もう安全！

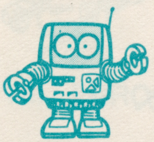
ところで、何回か説明したように、ファミコンでは、電源を切ってしまうと、せっかく入力したプログラムが全部消えてしまう。ベーシック・カセットに、電源を使ったバックアップ（保存）機能もあるが、残念ながら、ここにはBG画面を保存できないし、プログラムも１本しか入らない。



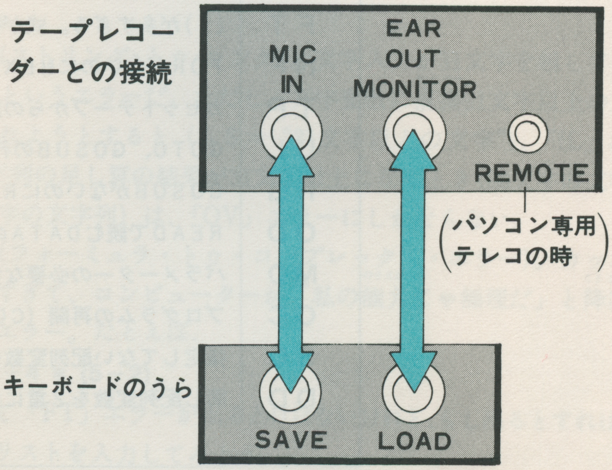
では、どうやってプログラムを保存するか。ファミコンでは、カセットテープにプログラムを保存することを「**SAVE**」（セーブ）すると言う。そのテープからプログラムをファミコンへ読み込むことを「**LOAD**」（ロード）すると言う。ベーシックの命令も、「SAVE」と「LOAD」だ。ファミコン専用のカセット・コレーダーも販売されているけれど、家庭用のラジカセや、ステレオのカセットデッキでも大丈夫だ(注)。さっそく、キーボードとテープレコーダーをつないでみよう。

**SAVE**

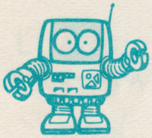
**LOAD**



**●図7 テープレコーダーとの接続**



接続は、図7のように。接続ケーブルは電機屋さんで売っている。「両端がミニジャックの**無抵抗線**」2本だ。SAVEとLOADは一度にはできないので、1本をそのたびにつなぎかえてもよい。



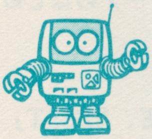
**SYSTEM**

Ver. 3では、BG画面とベーシック画面がすぐ切りかえられるけど、Ver. 2ではどうだろう。ダイレクトで、

**SYSTEM**

と打ち込もう。なつかしいGAME BASICのセレクト画面だ。「2」とキーインすると、BG画面になる。このBG画面でのLOAD・SAVE法は、前にも言った通りだ。35ページをもう一度見てください。

これが終わったら、ESCキーのあとSTOPキーを押して、ベーシック画面にもどる。



**TP　ERROR**

「F1」キーを押してみよう。「LOAD」と表示されたまま、カーソルも「OK」の表示も出ない。どのキーを押しても反応がない。STOPキーを押すと、

**?TP　ERROR**

と出て、やっと「OK」とカーソルが出てきた。

「F1」キーは「LOAD」と「CHR$ (13)」つまり、RETURNキーの5つの文字が定義されていた。そう、「LOAD」という命令をコンピューターに送っていたんだ。この命令を受けると、コンピューターはテープから読み出す仕事を始める。もしテープとつながっていなかったり、テープが入っていなかったりすると、STOPキーで中止の命令を出すまでは、ずーっとテープから信号が来るのを待っているんだ。

ところで、BG画面の所でも少し話したけど、テープに記録する時、他のプログラムやデーターと区別するために（このプログラムやデーターがテープに入ったものをファイルと呼ぶ)、ファイルネームを付けることができる。「SAVE」というのが、プログラムをテープへ保存せよという命令だから、ファイルネームを「TEST」と付ける時は、

**SAVE "TEST"**

とすればよい。「TEST」は文字列だから、「""」で囲むんだね。ということは、

**A$="TEST" : SAVE　A$**

でもよいわけだ。

**WRITING　TEST**

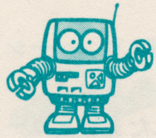
と表示して、プログラムをSAVEしてくれる。ファイルネームなんかいらない時は、

**SAVE**

だけでもよい。この時、ファイルネームは空白になる。ファイルネームは、文字変数と同じ31文字以内。それを越えると、「?IL ERROR」になる。

**LOAD　"TEST"**

とすると、「TEST」というファイル名をもつプログラムを探して、見つかったらLOADしろという命令になる。もし、テープのはじめに他のプログラムが入っていたら、



**SKIP**

**SKIP　XXX**

と表示して、「XXX」というプログラムを見つけたけど、「TEST」というファイルネームじゃないので、SKIP(無視して飛び越す)したわけだ。

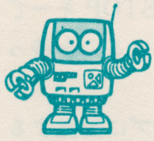
「F1」キーのように、単に、

**LOAD**

とだけすると、ファイルネームの空白なものを探せ、というのではなくて、何でもよいから一番最初に出てきたプログラムを呼び出せ、という命令になるんだ。

**SAVE　"TEST"**

としたあと、本当にちゃんと保存できたかな、と確かめるには、テープを巻きもどしたあと、



**LOAD?**

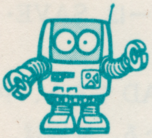
**LOAD?　"TEST"**

とする。単なる「LOAD」は、テープからプログラムを読み込んで、メモリーへ送る役目を持つけれど、「LOAD?」とすると、テープから読み込んだものを、今メモリーにあるものと比べるだけで、メモリーには送り込まない。「OK」が出ると、テープの中身とメモリーの中身が同じものというサインだ。違っていた時は、もう一度「SAVE」からやり直すわけだ。

BG画面のデーターの「SAVE」と「LOAD」は、まずBG画面を

呼び出さなければならないけれど、Ver. 3では、ベーシック画面で直接できる。

**LOADS**
**SAVES**



**LOADS "TESTBG"**
**SAVES "TESTBG"**

のように、プログラムの「LOAD」「SAVE」と区別するために、「S」を付けたものだ。もちろんVer. 2でセーブしたデーターを「LOADS」で読むこともできるよ。

普通のパソコンでは、テープを操作する時「リモート・コントロール」というのがあって、テープのモーター回転をコンピューターの方で止めたり、動かしたりできるけど、ファミコンにはそれがない。

**SAVEの時** ①SAVE "～"と入力。RETURNキーは押さない。
②録音（RECORD）ボタンを押して、テープを回す。
③RETURNキーを押す。

**LOADの時** ①LOAD "～"と入力。
②RETURNキーを押す。
③再生（PLAY）ボタンを押して、テープを回す。

という順番が確実だ。

テープレコーダーで再生ボリュームが自由に設定できるもの（ラジカセなど）は、ボリュームによっては、読み込めなくなるものもある。また、録音ボリュームも設定できるものも同じだ。

大事なプログラムを作る時に、どのテープレコーダーを使うか、そのレコーダーを使う時には、ボリュームをどのくらいにするか、何度もボリュームを動かして実験しておくべきだ。プログラムができ上がってから、予定していたレコーダーでは読み取れないとわかったら大変だからね。

すべてがOKなら、いよいよお待ちかね、プログラム作りだ。

大事なプログラムは、途中でこまめに何回もSAVEするくせをつけておこう。

〔注〕

我々の実験によると、普通のパソコン用テープレコーダーは、

SAVEに関してはOKだけど、LOADにはほとんど失敗した。ファミコンから送られる信号の電圧が低いせいだろう。ステレオ用のカセットデッキや、ラジカセの方がLOADできるようだ。

今、我々はSAVEにはパソコン用、LOADにはステレオ用デッキを接続しているんだ。ファミコン専用テープレコーダーは、SAVE・LOADとも問題はないから安心してよい。

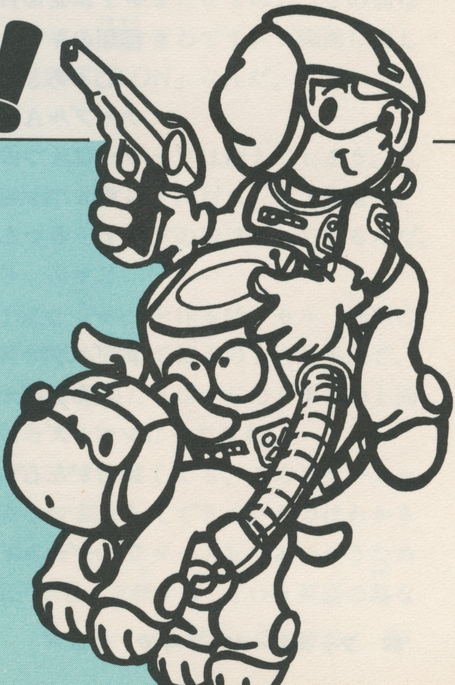
なお、Ver. 3でSAVEしたテープをVer. 2でLOADすると、中のプログラムが変わってしまうという現象が起きている。原因は、ただ今調査中。Ver. 2でSAVEしたものは、Ver. 3でも問題なくLOADできるようだ。(ステップ編で発表できる)

なお、ステレオ用のカセットデッキとファミコンをつなぐには、「ミニジャック」と「ミニプラグ」の無抵抗線を使う。

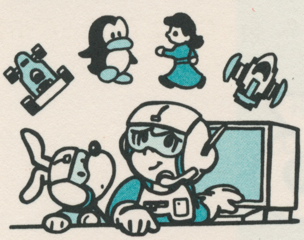
もし、LOADに失敗しても音量（ボリューム）をかえることで成功することもある。うまくいった時のボリュームのつまみの位置を覚えておこう。

# 作戦指令 ００３

# プログラムに挑戦だ！



# 003-1 キャラクターを表示しよう



いよいよ、GAME BASICを使ったゲームプログラムに挑戦だ。今まで、変数とか文字列とか、むずかしそうな話に付き合ってくれたて、ありがとう。今までこの本をまじめに1つずつキーインしてくれた君なら、ここからあとの所もすぐわかるはずだよ。

さて、このファミコンのベーシックの特長は、いろんなキャラクターがはじめからコンピューターに内蔵されているという点だ。巻末のキャラクターテーブルを見てみよう。普通のパソコンには、キャラクターテーブルAにあたるものはないので、ゲームのキャラクターは全部自分で作らなければならない。ファミコンには、この58種類のキャラクターが、みんな入っていて、自由に使えるんだ。

じゃあどうやってこれらのキャラクターを使うんだろう。それにテーブルAを見ればわかるように、たとえば「マリオ」のキャラクターは4つの記号で作られている。「PRINT CHR$（　）」として、この4つを組み合わせて1つのキャラクターを作るのはけっこうめんどくさいんだ。

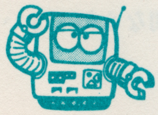
じゃあ、どうやってキャラクターを表示させるかというと、このベーシックでは2つの簡単な方法が用意されている。今回は、その内「スプライト機能」と呼ばれるものを使ってみよう。

どんなゲームを作ろうかな。テレビゲームブームのはじまりはインベーダー・ゲームだった。君たちは知っているかな。画面の下で砲台が左右に移動して、上から攻めてくるインベーダーをやっつける、というものだ。よし、初挑戦のゲームプログラムは、このタイプのゲームで行こう。

```
1Ø CLS
2Ø SPRITE ON
3Ø DEF SPRITE Ø,(Ø,1,Ø,Ø,Ø)=CHR$(172)+CHR$
(173)+CHR$(174)+CHR$(175)
5ØØ SPRITE Ø,11Ø,1ØØ
```

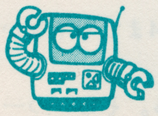
入力したら、「RUN」してみよう。画面のまん中あたりに、上向きのスターシップが表示されたかな。エラーが出たら、もう一度よく見てエディットしよう。

まず、10行の「CLS」。これはもう知ってるね。画面にリストの残りなんかが残っているとみっともないから、「RUN」すると同時に画面をきれいにしようというわけだ。



**SPRITE ON**

さて、20行の「**SPRITE ON**」。ファミコンには、バックグラウンド面（今までベーシックのリストを打ち込んだり、ダイレクトモードで使っていた所）の前に、スプライト面という画面を持っているんだ。そして「CGEN」という命令で変更しない限りは、キャラクターテーブルAのキャラクターは、全部このスプライト面で使われている。ところが、ふだんベーシックの命令を入力したりする時は、このスプライト面が表示されているとじゃまになるので、この面は消しているんだ。そこで、スプライト面を重ねてテレビ画面に出してくれ、という命令が、この「SPRITE ON」なんだ。



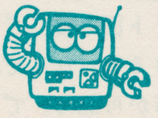
**SPRITE OFF**

```
SPRITE OFF
```

ダイレクトでこう命令すると、スターシップは消えてしまう。これは、スプライト面をテレビ画面に出すのと反対に、消す命令だ。でも、スプライト面に書かれたスターシップの絵は残っているんだ。

```
SPRITE ON
```

で、またスターシップが現われた。アニメを作る時、セル（透明のセルの上に絵を書いたもの）を何枚も重ねて使うのと同じ手法で、スターシップの絵が書かれたセルを重ねたりはずしたりするのと同じだ。「ON」「OFF」は、電気のスイッチにもあるね。



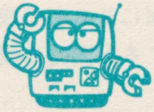
**DEF SPRITE**

さて次の「**DEF SPRITE**」だ。マリオにしても、このスターシップにしても、4つのキャラクターが集まってできているのはわかるね。この4つを1組にして1つのキャラクターとしてあつかうための命令なんだ。「DEF」は「DEFINE＝定義する」という英語の略な

んだ。つまり、

**DEF SPRITE Ø,(〜)=□**

ということでスプライト番号Øのキャラクターは、□の中のものを使って（〜）のようにする、とコンピューターに君に宣言しているわけだ。この番号は「Ø」というのは、ゲームには何個もキャラクターが登場するだろうし、そうするとそれぞれのキャラクターに番号を付けなければ、どれがどれだかわからなくなるからだ。そして、コンピューターは「Ø」から教え始めるというのも知っているね。この番号は、もちろん「Ø」から「7」の間なら何でもいいんだけどね。



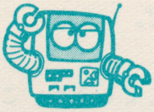
**SPRITE**

そして、5ØØ行としてある、

**SPRITE Ø, 11Ø, 1ØØ**

というのは、この番号Øのキャラクターを X＝11Ø、Y＝1ØØの所に表示しろ、という命令だ。最初を「DEF SPRITE 1、〜」としたら、もちろんここも、「SPRITE 1、〜」としなければいけないね。そして「11Ø、1ØØ」というのは、それぞれX座標、Y座標だということもわかるね。

ちょっと待てよ。X座標、Y座標はヨコ28、タテ24までじゃなかったっけ。



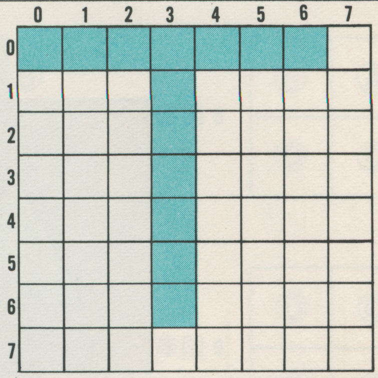
**ドット**

文字や数字は、実は8×8の小さな点に分解できる。その一つ一つが透明か、そうじゃないかでいろいろな文字を作っているんだ。たとえば「T」という文字は、図1のようになる。この一つ一つの点を「ドット」と言う。1文字＝8ドットというわけだ。だから今表示されているスターシップは、4つ分、つまり、タテ16ドット、ヨコ16ドットで作られているんだ。

キャラクターを動かす時、ヨコ28、タテ24を単位に動かすと、一度に8ドットも移動してしまう。だから、このスプライト面では、ドットを単位に表示する。そうすると細かな動きができるわけなんだ。

このX、Yは、それぞれØ〜255の数字を入れる事が可能だ。でも、28×8＝224だし、24×8＝192じゃないか。その通り。Ø〜255の数字ならば、例の「？IL ERROR」が起きない、というだけで、実際に意味を持つのはX＝Ø〜24Ø、Y＝5〜22Øの中だけだ。ためしにY＝255としてみると、スターシップは画面下ではなくて、上の

**●図1　文字のドット数**

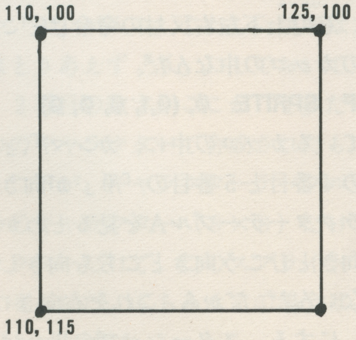


方にスターシップの下の部分だけが表示されてしまう。また、X＝11Ø、Y＝1ØØと指定すると、キャラクターが16×16の大きさだから、**図2**の場所に書かれるわけだ。つまり、ここで指定されるX座標、Y座標は、キャラクターの左上のドットの位置を意味するんだ。

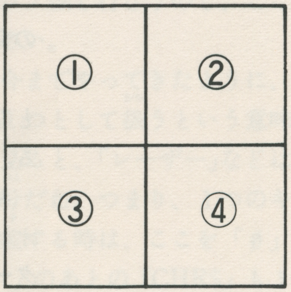
さて、もう一度「DEF　SPRITE」にもどろう。「＝」の右側に「CHR$（　）」が4つ並んでいる。これは、スプライト番号0のキャラクターは、172、173、174、175のコードナンバーで示された4つのキャラクターで作られている。という意味なんだ。この順番はどうでもよいかというと、そうではない。4つのキャラクターで構成されているということは、**図3**のようになっているはずなんだ。そして、たの順番（①～④）の通りに並べなければいけない。

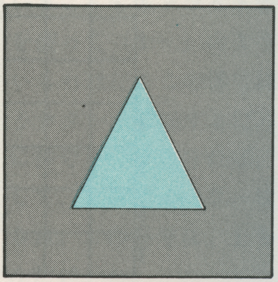
ところでこのスプイト機能、上下左右にひっくり返した表示も可

**●図2**

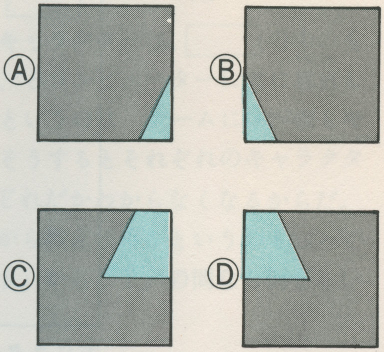



**●図3**





●図4




●図5

能なのだ。図4のようなキャラクターがあるとしよう。このⒶ～Ⓓをバラバラにすると図5になる。上向きのキャラクター（図4）とするときは、

**DEF　SPRITE　Ø,　(～)=CHR$ (Ⓐ)+CHR$ (Ⓑ)+CHR$ (Ⓒ)+CHR$ (Ⓓ)**

でよかったね。ここで、上下逆にするという命令を出すとどうなるだろう。Ⓐ～Ⓓは、それぞれ逆向きになるけど、それぞれの位置関係は前のままだから、図6のようになる。これを全体が逆向きにするためには、Ⓐ～Ⓓに並べかえて図7のようにしなければならない。そこで、

**DEF　SPRITE　Ø,　(～)=CHR$ (Ⓒ)+CHR$ (Ⓓ)+CHR$ (Ⓐ)+CHR$ (Ⓑ)**

と、なるわけだ。左右反対にする時も同じ考え方をすればわかるね。

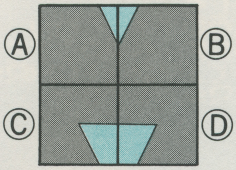
さて、その上下左右反対の命令はどこでするかというと、さっきの命令のカッコの中なんだ。

**DEF　SPRITE　Ø, (Ø,1, Ø, Ø, Ø)**

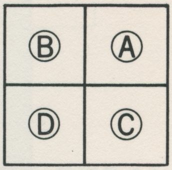
となっているカッコの中に、カンマ「,」で区切った5つの数字がある。この4番目と5番目の「Ø」が向きの指定なんだ。

キャラクターテーブルAを見るとわかるように、たとえば、マリオは左向きとむこう向きとこちら向きしかない。右向きのキャラクターがないんだ。だから、これを右向きにしたい時は、4番目の「Ø」を「1」にする。スターシップの場合は、上向きしかない。これを

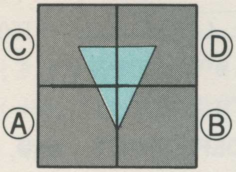
●図6



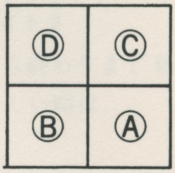

●図8




●図7




●図9




下向きにしたい時は、５番目の「∅」を「１」にすればよい。

つまり、この４番目の数字は左右逆にするか（「∅」で普通、「１」で左右逆）、５番目の数字は上下（「∅」で普通、「１」で上下逆）についての命令を意味するんだ。もちろん「＝」のあとの「CHR$」の順番はそれによって変わってくる。

**普通の時**　　ⒶⒷⒸⒹ　（図４）なら
**上下逆は**　　ⒸⒹⒶⒷ　（図７）
**左右逆は**　　ⒷⒶⒸⒹ　（図８）
**上下・左右とも逆は**　　ⒹⒸⒷⒶ　（図９）

のような順番になる。

さて残りの３つの数字だ。最初の数字はカラーを決める数字で「∅」～「３」が入る。３番目の数字は、表示する面を決めて「∅」か「１」が入る。くわしい説明はステップ編で行なうので、ここではとりあえず「∅」にしておいてください。

２番目の数字。ここが「１」だと、今までやってきたように、４つのキャラクターで合成されたものを１つとして扱うという意味だ。ところが、キャラクターテーブルAを見ると、「レーザー」などは１つのキャラクター（８×８）だけで充分だね。つまり、１つのキャラクターだけで、このスプライトを定義する時は、ここを「∅」にするわけだ。もちろんその場合は、「＝」のあとの「CHR$」も１個でよいわけだ。

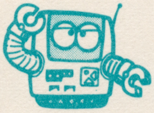
```
10 CLS
20 SPRITE ON
30 DEF SPRITE 0,(0,1,0,0,0)=
CHR$(172)+CHR$(173)+CHR$(174
)+CHR$(175)
```

さっそく、ここで残りのキャラクターも定義しておこう。敵はスターキラー（コードナンバー160～163）としよう。ただし、上から攻めてくるから上下を逆にして、スプライトナンバーを2にしよう。スプライト1は味方のレーザー、スプライト3は敵のレーザーだ。

**4Ø　DEF　SPRITE　1,(Ø, Ø, Ø, Ø, Ø)＝CHR$ (212)**

**5Ø　DEF　SPRITE　2,(Ø, 1, Ø, Ø, 1)＝CHR$ (162)＋CHR$(163)＋CHR$(16Ø)＋CHR$(161)**

ここまで入力したら、カーソルを40行まで持っていこう。「4」の所にカーソルを合わせて、「6」を入力して、カーソルを右に移動。「SPRITE 1」の「1」の所で「3」をキーインして、RETURN キーを押そう。



**LIST**

そのあとで、カーソルを下の方の何も書かれていない所へ持っていって、

**LIST　6Ø**

としてみよう。ちゃんと、

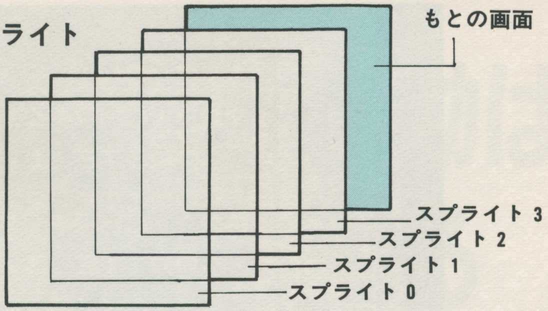
**6Ø　DEF　SPRITE　3,(Ø, Ø, Ø, Ø, Ø)＝CHR$ (212)**

と表示されたね。似たような内容の行は、こうやって簡単に入力できるんだ。行番号と、内容が違う所だけエディットして、そのあとRETURN キーを押せばよい。(40行はもとのままだから安心して！)

「LIST」の新しい使い方が出たね。今まで「LIST」のあと

```
40 DEF SPRITE 1,(0,0,0,0,0)=
CHR$(212)
50 DEF SPRITE 2,(0,1,0,0,1)=
CHR$(162)+CHR$(163)+CHR$(160
)+CHR$(161)
60 DEF SPRITE 3,(0,0,0,0,0)=
CHR$(212)
```

●図10　スプライト




RETURN キーを押せば、プログラムのはじめから終わりまでを全部見せてくれたけど、目的のプログラムをすぐ見たい時は、

**LIST　6Ø**

のように、行番号を付けて RETURN キーを押すと、その行だけを見せてくれる。また、

**LIST　4Ø−6Ø**

とすると、40行から60行までの間のプログラムだけを見せてくれる。

500行の一番左の数字を「Ø」から「3」まで変えてみると、それぞれのキャラクターが表示されるね。つまり、この「DEF SPRITE」と「SPRITE」の組み合わせで、君はキャラクターテーブルAのキャラクターを自由に表示できるようになったんだ。

**SPRITE　Ø**

と、X・Y座標(ざひょう)を指定しない時は、その番号のキャラクターは画面から消える。

また、同じナンバーのキャラクターをどこかに書くと、その前に書いていた場所の同じキャラクターの絵は消えていまう。つまり、スプライトという1枚のセルには、1つのキャラクターだけを書くと考えればよい。そのキャラクターを新しく書くということは、セルを移動するということになるから、前の場所のものが消えるということだ。(図10)

なお、このスプライトは、横(同じ行)に4枚以上重ねることはできないという制約がある。(スラップ編(へん)で説明(せつめい)する)

```
10 CLS
20 SPRITE ON
30 DEF SPRITE 0,(0,1,0,0,0)=
CHR$(172)+CHR$(173)+CHR$(174
)+CHR$(175)
40 DEF SPRITE 1,(0,0,0,0,0)=
CHR$(212)
50 DEF SPRITE 2,(0,1,0,0,1)=
CHR$(162)+CHR$(163)+CHR$(160
)+CHR$(161)
60 DEF SPRITE 3,(0,0,0,0,0)=
CHR$(212)
200 IF STICK(0)=1 THEN AX=AX
+1
210 IF STICK(0)=2 THEN AX=AX
-1

220 IF AX<0 THEN AX=0
230 IF AX>240 THEN AX=240
240 BY=BY+RND(2):BX=BX+RND(9
)-4
250 IF BX>240 THEN BX=240
260 IF BX<0 THEN BX=0
270 IF BY>220 THEN BY=5
700 GOTO 200
```



# 003-2

# 移動はどうする



もうわかるだろう。500行のX、Y座標を変えれば移動する。問題は、どうやってX、Yを変えるかだ。

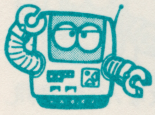
まず、スターシップを考えよう。スターシップは、いつも一番下を移動するわけだから、Y座標はいつも220と固定しよう。X座標は左右に動くわけだから、いつも変わっている。とすると変数を使うしかない。

**5ØØ　SPRITE　Ø, AX, 22Ø**

スターシップのX座標をAXという変数に入れたわけだ。他のも決めちゃおう。

**スターキラーのX, Y座標を　BX, BY**
**君のレーザーのX, Y座標を　CX, CY**
**敵のレーザーのX, Y座標を　DX, DY**

と決めよう。



**フラグ**

もう一つ、レーザーは発射した時だけ表示させるということにしなければならない。今、弾が出ているか出ていないかのマークが必要だ（このことを**フラグ**という）。

敵のレーザーが出ているかどうかをFB（0で出ていない。1で出ている）、味方のものをFA（0で出ていない。1で出ている）という変数に入れよう。

**52Ø　SPRITE　2, BX, BY**

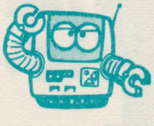
さて、レーザーは出ている時（つまり、フラグが1の時）だけ表示するわけだから、

**51Ø　IF　FA=1　THEN　SPRITE　1, CX, CY**
**53Ø　IF　FB=1　THEN　SPRITE　3, DX, DY**

とすればよい。IF 〜 THEN □ 文は覚えているかな。忘れてい

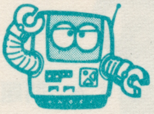
たら、もう一度58ページを見よう。

味方のスターシップを動かすにはどうすればよいだろう。コントローラーを使って右か左に動かすわけだから、コントローラーの十字のボタンが、右に押されているか、左に押されているかがわかればよいわけだ。



**STICK**

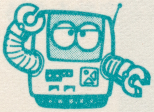
```
1 PRINT  STICK (Ø)
2 GOTO  1
```



**GOTO**

と打ち込んで「RUN」させてみよう。画面に次々と「Ø」が並んだね。1行の命令をすませると、コンピューターは2行へ進んで、「GOTO 1」という命令で、また1行の命令を実行する。

この「**GOTO**」という命令は、うしろにある数字の行番号の行の命令を実行しろ、という命令だ。分岐命令とも言う。もちろん、うしろに数字がなかったり、その数字の行番号を持つ行がなければ、エラーになる。



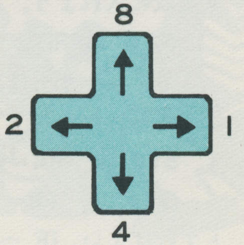
**無限ループ**

ここでは、コンピューターは1行と2行の間をずーっとくり返して、永遠に実行し続けるわけだ(このことを無限環、**無限ループ**という)。これを止めるには、STOPキーでプログラムの実行を止めるしかない。

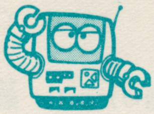
さて、まだ「Ø」の表示は続いているね。ここでコントローラーの十字ボタンを押してみよう。押している間だけ、「1」、「2」、「4」、「8」の数字が出たね。

1行の「STICK (Ø)」というのは、コントローラー1の十字ボタンをどっちの方角に押しているのかを知るための関数なんだ。押してない時は「Ø」、押している時は、図11の数字が返ってくる。この「STICK」のカッコの中の「Ø」を「1」にすると、コントローラー2の十字ボタンの押している方向がわかる。

●図11



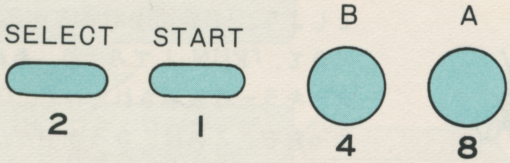
**STRIG**



**1 PRINT STRING (Ø)**

と変えてみる。また「Ø」が続いて、今度は十字ボタンを押しても数字が変わらない。この「STRING」は、コントローラーの他のボタンが押されているかを教えてくれる関数だ。押してない時は「0」だ。Aボタンを押してみよう。「8」になったね。Bボタンは「4」だ。AとBを一度に押すと「12」だ。12は8＋4で、2つ以上のボタンが押されると、それぞれの数値を足したものになる。図　の

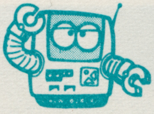
●図12




ように、それぞれのボタンの数値は割り当てられているんだ。

もちろん、この命令もカッコの中を「1」にすれば、コントローラー2に対する命令になるけど、コントローラー2には、STARTや、SELECTボタンはないから、返ってくる数値は、「Ø」「8」「4」「12」の4つだけだ。

さて、これで「STICK」と「STRING」はわかった。1行と2行を消そう。



**行を消す**

**1 「RETURN」キー**

**2 「RETURN」キー**

でリストをとると、1行2行が消えて、もとのリストが表示されたね。このように、行番号だけ入力して内容を書かずにRETURNキーを押すと、その行を消してしまうことができるのだ。

さて、コントローラーの十字ボタンが、右方向に押されていれば、スターシップを右に寄せればよいのだからAXの値をふやす。左方向ならへらせばいいね。

```
2ØØ  IF  STICK (Ø)=1  THEN  AX=AX+1
21Ø  IF  STICK (Ø)=2  THEN  AX=AX-1
```

ところで、X座標は0～240しか有効じゃないから、マイナスになるとエラーになる。このままだと、AXが0より少なくなることもあるし、240より大きくなることもある。

**22Ø IF AX<Ø THEN AX=Ø**

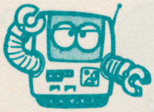
**23Ø IF AX>24Ø THEN AX=24Ø**

これで大丈夫だ。

ところで、スターシップは、一番下のまん中から始めたいし、敵のスラーキラーも一番上のまん中から始めたいね。でも、このプログラムのままでは、AXもBXも0から始まる。なぜって、どこにもAXやBXの最初の値（初期値）を決めていないからだ。そこで、

**1ØØ AX=12Ø：BX=12Ø：BY=5**

としよう。「BY=5」は、Y座標の有効範囲が5～22Øだからだ。



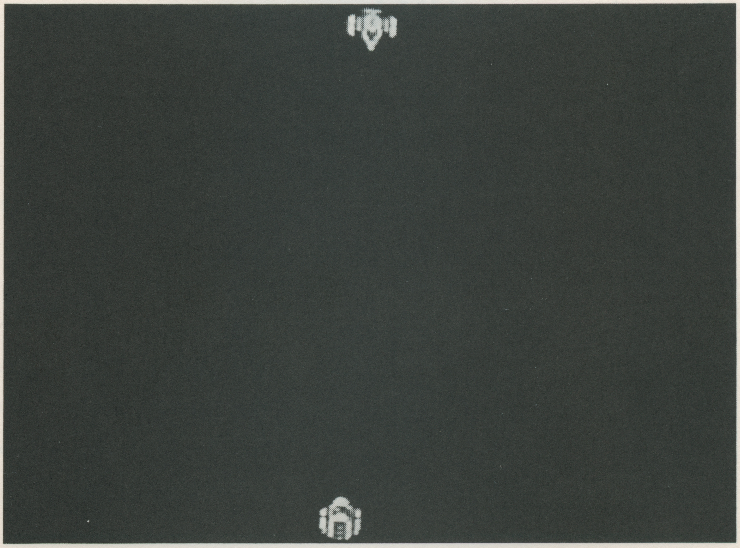
**くり返し**

さて、「RUN」してみよう。あれ、画面の上下にスターシップとスターキラーは表示されたけど、「OK」と出てプログラムが終了しちゃったぞ。これは、さっきやった「GOTO」のように、プログラムを何回もやれ、という命令がないから、1回通りすぎるとそれで終わりになったんだ。

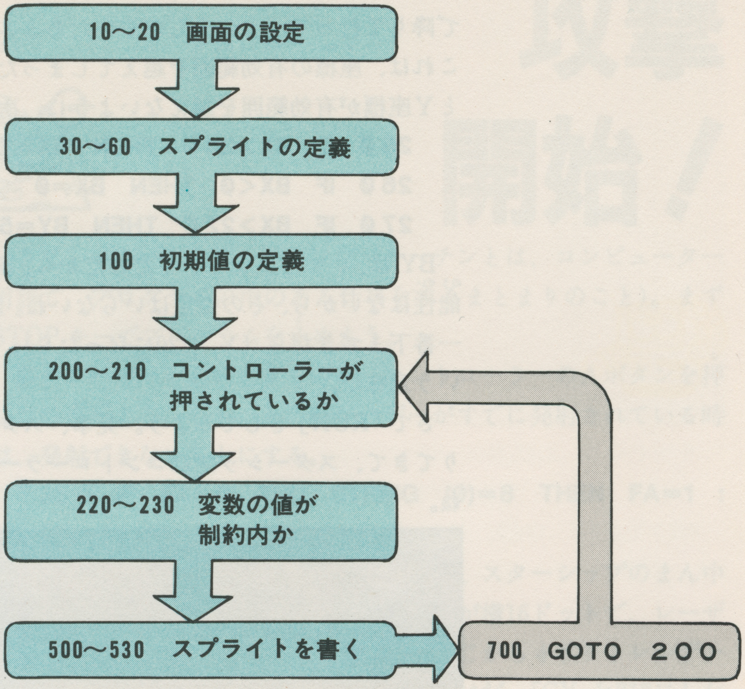
**7ØØ GOTO 2ØØ**

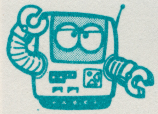
としてみよう。「RUN」させると「OK」は出なくなった。さあ、コントローラーを押してみよう。お！ 動いた、動いた。

それぞれのキャラクターを表示したあと、200行のコントローラーが押されているかどうかを見に行く所へ実行が移されている。図13



**●図13　プログラムの流れ**



**RND**

を見よう。プログラムの流れがよくわかるね。

今度は敵の動きだ。

**24∅　BY＝BY＋RND (2) : BX＝BX＋RND (9)−4**

新しい単語「RND」だ。これは、でたらめな数（乱数）を作るための関数だ。ゲームをやっている時、いつも敵が同じような動きをしたんじゃ、パターンがわかってしまってつまらない。そこで、この乱数を使って、わざとでたらめな動きをさせるのだ。

「A＝RND (1)」だと、Aはいつも０、「A＝RND (2)」だと、Aは０か１、「A＝RND (3)」では、Aは０か１か２、というわけだ。たとえば、100から110までの11通りの乱数を作りたい時は、

**A＝RND (11)＋1∅∅**

というようにするわけだ。

だから、240行では、BYは０から１かずつふえ、BYは−４から＋

4の範囲で、ふえたりへったりするわけだ。

ここでまた「RUN」してみよう。スターキラーがふらふらと下まで降りて行って、上に少しのぞいたあとエラーが出て止まったね。これは、座標の有効範囲を越えてしまったからだ。そこで、X座標とY座標が有効範囲を越えないように、次の行を入れる。

```
25Ø  IF  BX>24Ø  THEN  BX=24Ø
26Ø  IF  BX<Ø  THEN  BX=Ø
27Ø  IF  BX>22Ø  THEN  BY=5
```

BYは、いつもふえていくだけだから、BYが0より小さくなる可能性はないから、その指定はいらないね。270行は、スターキラーが一番下まできたら（Y>220になったら）、一番上に持っていく、という意味だ。

さて「RUN」としてみよう。見事、スターキラーはふらふらと降りてきて、スターシップはコントローラーで自由に左右に動かせるね。



# 003-3

# 攻撃開始！



さて、いよいよ攻撃ルーチンだ（ルーチンとは、コンピューター用語で、プログラムの中のある仕事をするまとまりのこと）。まずSTOPキーでプログラムを止めよう。

君がレーザーを発射できるのは、コントローラーのAボタンを押した時にしよう。でも、その時レーザーがすでに発射されている時は、発射できないことにする。

```
35Ø IF FA=Ø AND STRING (Ø)=8 THEN FA=1 : CX=AX+4 : CY=212
```

「CX」をAX＋4としたのは、レーザーが、スターシップのまん中から出るようにするためだ。スターシップが横16ドットで、レーザーが8ドットだから、スターシップの左はじから4ドットレーザーをずらすことで、ちょうどまん中になるわけだ。「CY」を212にしたのも、やはり、スターシップとレーザーが重ならないように8ドット分引いたのだ。

さて、スターキラー側のレーザーはどうしよう。当然スターキラー側のレーザーが出てない（FB＝Ø）時に1／6の確率で発射できるようにしよう。

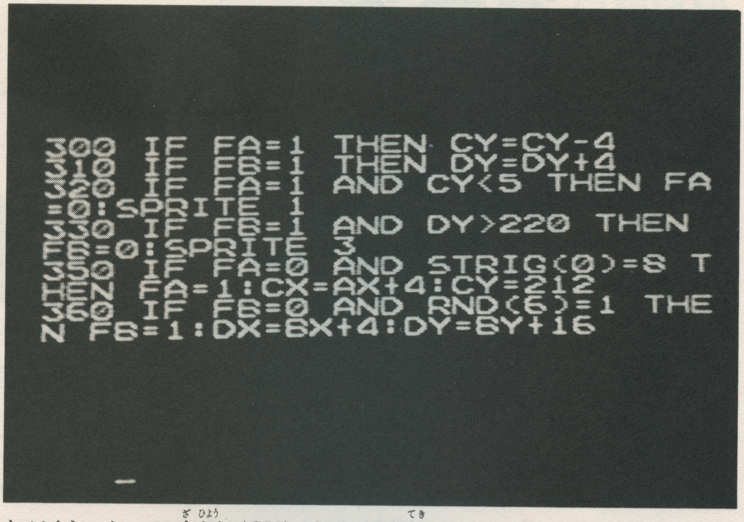
```
36Ø IF FB=Ø AND RND (6)=Ø THEN FB=1 : DX=BX+4 : DY=BY+16
```

「DX」は前と同じ。「DY」はスターキラーが、タテ16ドットだから、スターキラーのY座標に16ドット足すわけだ。

さて、この両方を動かさなくてはならない。

```
3ØØ IF FA=1 THEN CY=CY-4
31Ø IF FB=1 THEN DY=DY+4
```

レーザーは、1回に4ドット動かす。味方のレーザーは上に動く




わけだから、Y座標（CY）を－4、敵のレーザーは下に動くから、Y座標（DY）は＋4だ。もちろん、フラグ（FAとFB）が「Ø」だったら動かしてはいけないから、IF文でやらなくてはだめだ。

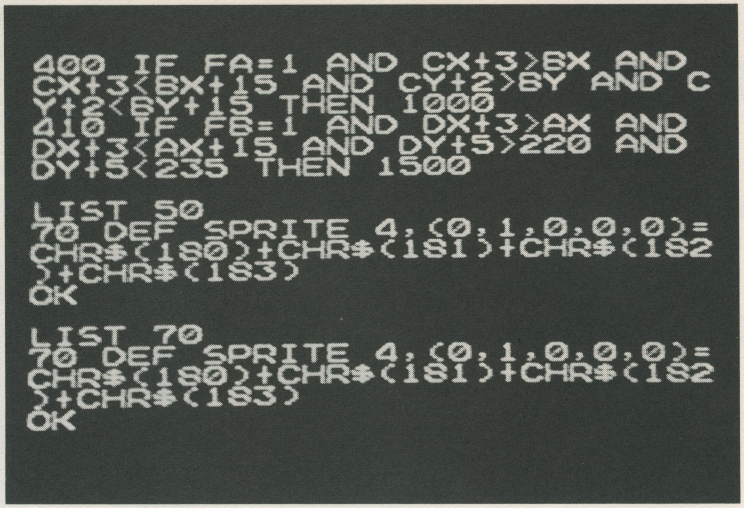
さて、また制限範囲かどうかを調べて、同時に上か下へ行きぬけたら、そのレーザーを消さなければならない。

**32Ø IF FA=1 AND CY<5 THEN FA=Ø : SPRITE 1**
**33Ø IF FB=1 AND DY>22Ø THEN FB=Ø : SPRITE 3**

「Y」は少なくなるだけだから、「CY>22Ø」は調べなくてもよい。「CY<5」になれば、レーザーが出ているかどうかのヘラグ（FA）を「Ø」にして、同時にレーザーを消す（SPRITE 1)。敵のレーザーも、まったく同じ処理だ。

この300行から330行を、さっきの350・360行の手前に置いたのは、発射した直後にここを通ると、レーザーが1回分だまって通りすぎてしまうからだ。

さて今度は、それぞれのレーザーが相手に当たったかどうかの判定だ。話を簡単にするため、レーザーが当たったかの判定は、レーザーの先頭の部分が相手のキャラクターの中へ入ったかどうかで調べよう。レーザーの先頭は、味方のをCX＋3、CY＋2として、敵の部分はDX＋3、DY＋5としよう。Y座標の数値が違うのは、CY




は上向き、DYは下向きで、それにレーザーの先頭は、そのキャラクターの書かれている一番上の部分より少し内側だからだ。

**4ØØ IF FA=1 AND CX+3>BX AND CX+3<BX+15 AND CY+2>BY AND CY+2<BY+15 THEN 1ØØØ**

**41Ø IF FB=1 AND DX+3>AX AND DX+3<AX+15 AND DY+5>22Ø AND DY+5<235 THEN 15ØØ**

400行、410行で、それぞれ当たっていたと判定された場合は、敵をやっつけたのなら1000行へ、やられたなら1500行へ飛べ、というわけだ。

「IF ～ THEN」の、「THEN」のうしろは命令だったはずだけど、数字がある場合にはその行へ飛べ、という意味になる。これは、

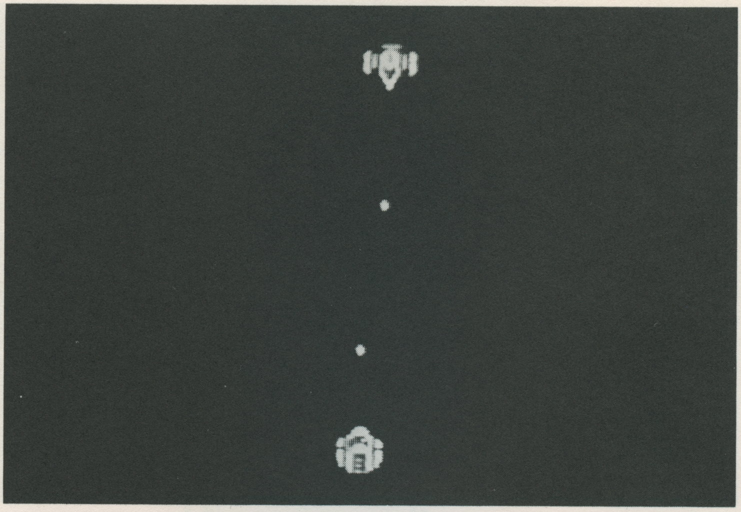
**IF ～ THEN GOTO 1ØØØ**

というのが、いつの間にか「GOTO」が省略されるようになったものだ。

さて、ファミコンのキャラクターには、爆発の絵もあったね。

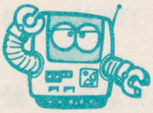
**7Ø DEF SPRITE 4, (Ø, 1, Ø, Ø, Ø)=CHR$ (18Ø)+CHR$ (181)+CHR$ (182)+CHR$ (183)**

を追加しよう。



そして、1000行と1500行では、それぞれのレーザーの絵とフラグを消して、やられた船を消して、かわりに爆発のキャラクターを書けばよい。

```
1ØØØ  FA=Ø : SPRITE  1 : SPRITE  2
1Ø1Ø  SPRITE  4, BX, BY
11ØØ  GOTO  2ØØØ
15ØØ  FB=Ø : SPRITE  2 : SPRITE Ø
151Ø  SPRITE  4, AX, 22Ø
2ØØØ  END
```

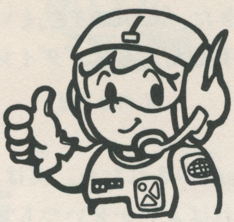


**END**

「END」は、これでプログラムは終わり、という命令。この場合は、付けなくてもいいけど、あとで、「サブルーチン」という手法を使うようになると、リストの途中でプログラムを終わらせることもあるので、使う習慣をつけた方がいいよ。

さあ、これで「RUN」してみよう。入力ミスさえなければ、これで立派に動くはずだ!!

# 003-4



# これでゲームは完成だ！

でき上がったゲームの感想はどうかな。何だか物足りない？。原因は、どっちかの船が１回やられるとそれで終わってしまうからだ。それじゃあ、もう少し手を加えてみよう。次の行を加えてください。

```
90    SC=0 : SH=3
1020  SC=SC+10
1520  SH=SH−1
```

そして、2000行以下を次のように変えよう。

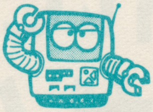
```
2000  LOCATE  10, 5 : PRINT  "SCORE  =" ; SC
2010  IF  SH=0  THEN  3000
2020  LOCATE  10, 8 : PRINT  "ノコリ  =" : SH
2100  BEEP
2110  FOR  I=0  TO  1500 : NEXT
2120  BEEP : SPRITE  4
2200  CLS : GOTO  100
3000  LOCATE  10, 8 : PRINT  "GAME  OVER!"
3100  END
```

また、100行は、

100　AX=120：BX=120：BY=5：FA=0：FB=0

と直そう。

**LOCATE**



新しい命令が5つ、新しい記号が1つ出てきた。「LOCATE」というのは、「〜の場所を定める」という意味の英語だ。今までダイレクトモードで、

PRINT　"ABC"

などと命令すると、そのすぐ下に「ABC」と表示されたね。実は、コンピューターは、次にどこに文字を表示すればよいかをいつも覚えていて、「PRINT」文は、そこに文字を書け、という命令だったんだ。ところが「LOCATE」という命令を使うと、次にどこの位置から書き始めるかを、君が決める命令なのだ。たとえば、

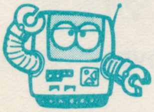
LOCATE 0, 4：PRINT　"ABC"

と書くと、X座標＝0，Y座標＝4の場所に「A」を書けという意味だ。もちろん「BC」は「A」のすぐうしろに書かれているね。だから、「LOCATE」のあとの2つの数字は、それぞれX座標とY座標を表わして、Xは0〜27、Yは0〜23の間じゃないとエラーになる。

PRINT　"A"：PRINT　"B"

PRINT　"A"；：PRINT　"B"

この2つをためしてみよう。上の命令では、2行にわたって、「A」、「B」と表示されたのに、下の命令では、「AB」と1行に続けて表示されたね。「PRINT 〜」として「〜」にあたる文字列、数値、変数のあとに何もない時は、この行にはここまでしか書かないから、次のPRINT文が出てきたら、下の行に送れ（改行＝行を改ためる）、という意味になるんだ。



**;**

ところが、うしろに「；」（セミコロン）があると、行をかえないで、続けて書いて、という意味になる。

**BEEP**

「BEEP」。この命令があると、コンピューターが「ビーッ」という短いブザー音を出す。

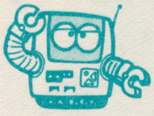
**FOR**
**TO~**
**NEXT**

残りの3つ、「FOR」と「TO」と「NEXT」。これは、本当はもう一つ「STEP」という命令をふくめて4個1組の命令なんだ。

FOR　I=0　TO　500：NEXT

これは

FOR　I=0　TO　500　STEP　1：NEXT



STEP 2

という文型の省略（STEP 1を省略）したものだ。「I」という変数に、まずØという値を入れて、そのあと、Iの値を1ずつ増していって、Iが500になるまでそれをくり返す。というのが上の例の意味だ。

増していく値（増分値）が2づつの時は、「STEP 2」のように書く。普通ここが1の時は、このプログラムのように「STEP 1」を省略することができるんだ。

このプログラムでは、Iを0から1500まで数えている間、コンピューターは他の仕事ができないから、時間待ちとして使っているけれど、この「FOR ～ NEXT」文は、プログラムの中でも、重要な命令なんだ。

ダイレクトでできる簡単な例で、

**A=Ø : FOR I=1 TO 1ØØ : A=A+I : NEXT : PRINT A**

これは1から100までの数を全部足して、その答えを出せ、という命令がこんな簡単にできる。

**A=Ø : A=A+1 : A=A+2 : A=A+3 : ～ : A=A+99 : A=A+1ØØ : PRINT A**

と同じ意味だ。どっちがいいかな。なぜそうなるかは、ステップ編でお話ししましょう。

変数「SC」は得点で、「SH」は船の数だということはすぐわかるね。90行と100は両方とも初期値だけど、90行はゲーム全体の初期値、100行は1回戦の分の初期値と分けている（初期値とは、読んで字のごとく、一番初めに決めておく数値）。

最終リストは次のページの通りだ。

どうだい。わずか14個の命令だけで、こんなゲームが簡単にできた。いろんな所の数値を少し変えるだけで、このゲームはいくらでも改造できる。

さあ、これに続く「ステップ編」では、もっと命令を覚えて、迷路ゲームに挑戦だ。

```
10 CLS
20 SPRITE ON
30 DEF SPRITE 0,(0,1,0,0,0)=CHR$(172)+CHR$(173)+CHR$
(174)+CHR$(175)
40 DEF SPRITE 1,(0,0,0,0,0)=CHR$(212)
50 DEF SPRITE 2,(0,1,0,0,1)=CHR$(162)+CHR$(163)+CHR
$(160)+CHR$(161)
60 DEF SPRITE 3,(0,0,0,0,0)=CHR$(212)
70 DEF SPRITE 4,(0,1,0,0,0)=CHR$(180)+CHR$(181)+CHR$
(182)+CHR$(183)
90 SC=0:SH=3
100 AX=120:BX=120:BY=5
200 IF STICK(0)=1 THEN AX=AX+1
210 IF STICK(0)=2 THEN AX=AX-1
220 IF AX<0 THEN AX=0
230 IF AX>240 THEN AX=240
240 BY=BY+RND(2):BX=BX+RND(9)-4
250 IF BX>240 THEN BX=240
260 IF BX<0 THEN BX=0
270 IF BY>220 THEN BY=5
300 IF FA=1 THEN CY=CY-4
310 IF FB=1 THEN DY=DY+4
320 IF FA=1 AND CY<5 THEN FA=0:SPRITE 1
330 IF FB=1 AND DY>220 THEN FB=0:SPRITE 3
350 IF FA=0 AND STRIG(0)=8 THEN FA=1:CX=AX+4:CY=212
360 IF FB=0 AND RND(6)=1 THEN FB=1:DX=BX+4:DY=BY+16
400 IF FA=1 AND CX+3>BX AND CX+3<BX+15 AND CY+2>BY A
ND CY+2<BY+15 THEN 1000
410 IF FB=1 AND DX+3>AX AND DX+3<AX+15 AND DY+5>220
AND DY+5<235 THEN 1500
500 SPRITE 0,AX,220
510 IF FA=1 THEN SPRITE 1,CX,CY
520 SPRITE 2,BX,BY
530 IF FB=1 THEN SPRITE 3,DX,DY
700 GOTO 200
1000 FA=0:SPRITE 1:SPRITE 2
1010 SPRITE 4,BX,BY
1020 SC=SC+10
1100 GOTO 2000
1500 FB=0:SPRITE 3:SPRITE 0
1510 SPRITE 4,AX,220
1520 SH=SH-1
2000 LOCATE 10,5:PRINT "SCORE =";SC
2010 IF SH=0 THEN 3000
2020 LOCATE 10,8:PRINT "ノコリ =";SH
2100 BEEP
2110 FOR I=0 TO 1500:NEXT
2120 BEEP:SPRITE 4
2200 CLS:GOTO 100
3000 LOCATE 10,8:PRINT "GAME OVER !"
3100 END
```

# ゲームキャラクター表(キャラクターテーブルA)



# キャラクターコード表

| コード | 文字 | コード | 文字 | コード | 文字 | コード | 文字 |
|---|---|---|---|---|---|---|---|
| 0 | (A0) | 32 | | 64 | @ | 96 | ア |
| 1 | (A1) | 33 | ! | 65 | A | 97 | イ |
| 2 | (A2) | 34 | ″ | 66 | B | 98 | ウ |
| 3 | (A3) | 35 | # | 67 | C | 99 | エ |
| 4 | (A4) | 36 | $ | 68 | D | 100 | オ |
| 5 | (A5) | 37 | % | 69 | E | 101 | カ |
| 6 | (A6) | 38 | & | 70 | F | 102 | キ |
| 7 | (A7) | 39 | ′ | 71 | G | 103 | ク |
| 8 | (B0) | 40 | ( | 72 | H | 104 | ケ |
| 9 | (B1) | 41 | ) | 73 | I | 105 | コ |
| 10 | (B2) | 42 | * | 74 | J | 106 | サ |
| 11 | (B3) | 43 | + | 75 | K | 107 | シ |
| 12 | (B4) | 44 | , | 76 | L | 108 | ス |
| 13 | (B5) | 45 | − | 77 | M | 109 | セ |
| 14 | (B6) | 46 | . | 78 | N | 110 | ソ |
| 15 | (B7) | 47 | / | 79 | O | 111 | タ |
| 16 | (C0) | 48 | 0 | 80 | P | 112 | チ |
| 17 | (C1) | 49 | 1 | 81 | Q | 113 | ツ |
| 18 | (C2) | 50 | 2 | 82 | R | 114 | テ |
| 19 | (C3) | 51 | 3 | 83 | S | 115 | ト |
| 20 | (C4) | 52 | 4 | 84 | T | 116 | ナ |
| 21 | (C5) | 53 | 5 | 85 | U | 117 | ニ |
| 22 | (C6) | 54 | 6 | 86 | V | 118 | ヌ |
| 23 | (C7) | 55 | 7 | 87 | W | 119 | ネ |
| 24 | (D0) | 56 | 8 | 88 | X | 120 | ノ |
| 25 | (D1) | 57 | 9 | 89 | Y | 121 | ハ |
| 26 | (D2) | 58 | : | 90 | Z | 122 | ヒ |
| 27 | (D3) | 59 | ; | 91 | 「 | 123 | フ |
| 28 | (D4) | 60 | 〈 | 92 | ¥ | 124 | ヘ |
| 29 | (D5) | 61 | = | 93 | 」 | 125 | ホ |
| 30 | (D6) | 62 | 〉 | 94 | ^ | 126 | マ |
| 31 | (D7) | 63 | ? | 95 | — | 127 | ミ |

| コード | 文字 | コード | 文字 | コード | 文字 | | コード | 文字 | |
|---|---|---|---|---|---|---|---|---|---|
| 128 | ム | 160 | ゾ | 192 | (F0) | マリオ用 | 224 | (J0) | 迷路 |
| 129 | メ | 161 | ダ | 193 | (F1) | | 225 | (J1) | |
| 130 | モ | 162 | ヂ | 194 | (F2) | | 226 | (J2) | |
| 131 | ヤ | 163 | ヅ | 195 | (F3) | | 227 | (J3) | |
| 132 | ユ | 164 | デ | 196 | (F4) | | 228 | (J4) | |
| 133 | ヨ | 165 | ド | 197 | (F5) | | 229 | (J5) | |
| 134 | ラ | 166 | バ | 198 | (F6) | | 230 | (J6) | |
| 135 | リ | 167 | ビ | 199 | (F7) | | 231 | (J7) | |
| 136 | ル | 168 | ブ | 200 | (G0) | LANDSCAPE | 232 | (K0) | グラフ用ケイ線 |
| 137 | レ | 169 | ベ | 201 | (G1) | | 233 | (K1) | |
| 138 | ロ | 170 | ボ | 202 | (G2) | | 234 | (K2) | |
| 139 | ワ | 171 | パ | 203 | (G3) | | 235 | (K3) | |
| 140 | ン | 172 | ピ | 204 | (G4) | | 236 | (K4) | |
| 141 | ヲ | 173 | プ | 205 | (G5) | 星(1) | 237 | (K5) | |
| 142 | ァ | 174 | ペ | 206 | (G6) | 星(2) | 238 | (K6) | |
| 143 | ィ | 175 | ポ | 207 | (G7) | ポール | 239 | (K7) | |
| 144 | ゥ | 176 | □ | 208 | (H0) | ドンキーコングJr.用 | 240 | (L0) | |
| 145 | ェ | 177 | 。 | 209 | (H1) | | 241 | (L1) | |
| 146 | ォ | 178 | [ | 210 | (H2) | | 242 | (L2) | |
| 147 | ャ | 179 | ] | 211 | (H3) | | 243 | (L3) | ボード用ワク |
| 148 | ュ | 180 | © | 212 | (H4) | | 244 | (L4) | |
| 149 | ョ | 181 | × | 213 | (H5) | | 245 | (L5) | |
| 150 | ッ | 182 | ÷ | 214 | (H6) | | 246 | (L6) | |
| 151 | ガ | 183 | ファ | 215 | (H7) | | 247 | (L7) | |
| 152 | ギ | 184 | (E0) | 216 | (I0) | 島 | 248 | (M0) | インクツボ |
| 153 | グ | 185 | (E1) | 217 | (I1) | | 249 | (M1) | |
| 154 | ゲ | 186 | (E2) | 218 | (I2) | | 250 | (M2) | |
| 155 | ゴ | 187 | (E3) | 219 | (I3) | | 251 | (M3) | |
| 156 | ザ | 188 | (E4) | 220 | (I4) | | 252 | (M4) | |
| 157 | ジ | 189 | (E5) | 221 | (I5) | 迷路 | 253 | (M5)(1) | ヌリツブシ |
| 158 | ズ | 190 | (E6) | 222 | (I6) | | 254 | (M6)(2) | |
| 159 | ゼ | 191 | (E7) | 223 | (I7) | | 255 | (M7)(3) | |

検印省略 NDC 548

ファミリーベーシック ホップ

昭和60年5月27日 発行

定価 800円

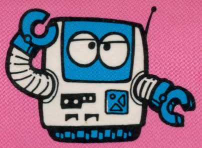
著者 地球防衛軍
発行者 小川茂男
発行所 (株)誠文堂新光社
東京都千代田区神田錦町1—5—5
郵便番号 101
電話 東京 (292) 1211
振替口座 東京 7—6294
印刷・広研印刷株式会社
製本・岡嶋製本工業



ISBN4-416-18516-2 C2055


FAMILY BASIC

ファミコン大作戦
ファミリーベーシック™ホップ



ファミコン大作戦

誠文堂新光社　ISBN4-416-18516-2 C-2055 ¥800E　定価 800 円



ファミコン大作戦 ファミリーベーシック™ ホップ
誠文堂新光社